George Feuerlicht
Winfried Lamersdorf (Eds.)

LNCS 5472

Service-Oriented Computing – ICSOC 2008 Workshops

ICSOC 2008 International Workshops
Sydney, Australia, December 2008
Revised Selected Papers

Springer

Lecture Notes in Computer Science 5472

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

George Feuerlicht
Winfried Lamersdorf (Eds.)

Service-Oriented Computing – ICSOC 2008 Workshops

ICSOC 2008 International Workshops
Sydney, Australia, December 1, 2008
Revised Selected Papers

Springer

Volume Editors

George Feuerlicht
Department of Information Technology
University of Economics Prague, Czech Republic
E-mail: jirif@vse.cz

and

Faculty of Engineering and Information Technology
University of Technology
Sydney, NSW 2007, Australia
E-mail: jiri@it.uts.edu.au

Winfried Lamersdorf
Department of Informatics
Distributed and Information Systems
University of Hamburg
22527 Hamburg, Germany
E-mail: lamersdorf@informatik.uni-hamburg.de

Library of Congress Control Number: Applied for

CR Subject Classification (1998): C.2, D.2, D.4, H.4, H.3, K.4.4

LNCS Sublibrary: SL 2 – Programming and Software Engineering

ISSN 0302-9743
ISBN-10 3-642-01246-9 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-01246-4 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12653959 06/3180 5 4 3 2 1 0

Preface

The International Conference on Service-Oriented Computing (ICSOC) has been tracking the developments in service-oriented computing since its early stages with the first conference held in Trento, Italy in 2003. Today, service-oriented computing has become a mainstream computing paradigm with an increasing number of organizations adopting service-oriented architecture (SOA). While the main ICSOC conference focuses on the core issues in service-oriented computing, ICSOC workshops have played a major role in providing a forum for the discussion of emergent research topics. Starting in 2005 at the Amsterdam conference, workshops have been an integral component of the ICSOC conferences. While the number of workshops and the topics covered change from year to year, the quality of the workshop publications has steadily increased with the most recent workshop proceedings published in the LNCS series by Springer.

Reflecting the growing interest in the area of service-oriented computing and the expanding field of application of software services, the ICSOC 2008 conference in Sydney, Australia was accompanied by five workshops focusing on specific research areas:

- Forth International Workshop on Engineering Service-Oriented Applications (WESOA 2008)
- Second International Workshop on Web APIs and Services Mashups (Mashups 2008)
- First International workshop on Quality-of-Service Concerns in Service Oriented Architectures (QoSCSOA 2008)
- First International Workshop on Enabling Service Business Ecosystems (ESBE 2008)
- Third Workshop on Trends in Enterprise Architecture Research (TEAR 2008)

The 2008 workshops covered well-established areas of research such as the engineering of service-oriented applications (WESOA 2008), as well as emerging topics that include Web APIs and services mashups (Mashups 2008) and quality-of-service concerns (QoSCSOA 2008), and provided a forum for the exchange of ideas between researchers and practitioners in the specific research areas.

The WESOA 2008 workshop in its fourth year addressed the challenges that arise from the unique characteristics of service-oriented applications, focusing in particular on service-oriented analysis and design and key principles that underpin the entire service-oriented development lifecycle. The workshop featured nine research papers, with several authors discussing fundamental SOA principles including unified management and governance and SOA. Other authors proposed service system engineering methods for simulation, automatic testing, and behavioral analysis of SOA processes. Finally, WESOA 2008 included

contributions on software service design for specific application scenarios such as human-based e-services, service value networks, and scientific workflows.

The theme of the second Mashups workshop (Mashups 2008) was convergence of service-oriented computing and Web 2.0 with specific focus on mashups and end-user-oriented compositions of Web-accessible services. The Mashups workshop series provides a forum for the discussion of research issues and challenges that emerge as a result of the interaction of Web 2.0 approaches and technologies with the existing service-oriented computing solutions that have to be addressed in order to achieve the vision of a fully programmable Web. The workshop program featured two invited keynote presentations and five research papers with topics ranging from social aspects of mashup applications and the analysis of mashup ecosystems to the discussion of the impact of mashup technology on organizations, including a demonstration of visual mashup development tools.

The Quality-of-Service Concerns in Service-Oriented Architectures(QoSCSOA 2008) workshop was the first ICSOC workshop focusing on quality-of-service (QoS) issues in service-oriented computing, covering the entire lifecycle of SOA-based systems. This half-day workshop included a keynote presentation and five research papers with topics ranging from the discussion of challenges of assuring QoS in service-oriented environments to technical papers discussing redundancy protocol for SOA, and trust models for service-oriented multi-agent systems. With the increasing adoption of SOA solutions there is growing interest in ensuring that application systems meet non-functional QoS requirements, making this workshop a valuable addition to the ICSOC workshop series.

The Enabling Service Business Ecosystems (ESBE 2008) workshop was another new ICSOC workshop that focuses on the problem of service development across business domains. The workshop included six papers with the topics covering description of services for service ecosystems, discussing service selection in business service ecosystem, service standardization, and analysis of evolutionary changes of Web services.

The Third Workshop on Trends in Enterprise Architecture Research (TEAR 2008) is a continuation of a workshop series on enterprise architectures, with the 2008 workshop focusing specifically on SOA. The eight research papers presented at the workshop covered topics on governance aspects of SOA, extensions of enterprise models, and common terminology for describing enterprise architectures.

The ICSOC 2008 workshops were very well received by the ICSOC attendees with more than 120 delegates participating in the five workshops. Following the highly successful workshop events on December 1, 2008, the workshop organizers were instrumental in compiling the papers for the individual workshops and ensuring that the reviewers' recommendations were incorporated into the final camera-ready versions. This process of finalizing and compiling the paper contribution was completed in January 2009, ensuring that the workshop proceedings could be published without excessive delays. ICSOC 2008 workshop proceedings is a compilation of partially revised versions of papers from all five workshops published as one book integrating the contributions into a single

unified structure, giving the readers access to an extensive range of the most recent research papers in this fast evolving area.

Our sincere thanks go to all workshop organizers: Yen-Jao Chung, Wolfgang Emmerich, Guadalupe Ortiz and Christian Zirpins for the WESOA 2008 workshop, Michael Maximilian, Stefan Tai, Cesare Pautasso and Patrick Chanezon for the Mashups 2008 workshop, Liam O'Brien and Paul Brebner for the QoSCSOA 2008 workshop, Vincenzo D'Andrea, G.R. Gangadharan Renato Iannella, and Michael Weiss for the ESBE 2008 workshop, and Pontus Johnson, Joachim Schelp and Stephan Aier for the TEAR 2008 workshop - and to Kai Jander who helped with formatting this book.

Sydney, Hamburg, Prague
February 2009

George Feuerlicht
Winfried Lamersdorf

Organization

ICSOC 2008 Workshop Chairs

George Feuerlicht	University of Economics, Prague, Czech Republic
Winfried Lamersdorf	University of Hamburg, Germany

WESOA 2008 Organizers

Jen-Yao Chung	IBM T.J. Watson Research Center, USA
Wolfgang Emmerich	University College London, UK
Guadalupe Ortiz	University of Extremadura, Spain
Christian Zirpins	University of Karlsruhe, Germany

Mashups 2008 Organizers

Michael Maximilien	IBM Almaden Research Center
Stefan Tai	University of Karlsruhe, Germany
Cesare Pautasso	University of Lugano, Switzerland
Patrick Chanezon	Google Inc., San Francisco, USA

QoSCSOA 2008 Organizers

Liam O'Brien	National ICT Australia, Canberra, Australia
Paul Brebner	National ICT Australia, Canberra, Australia

ESBE 2008 Organizers

Vincenzo D'Andrea	University of Trento, Italy
G.R. Gangadharan	University of Trento, Italy
Renato Iannella	National ICT Australia, Brisbane, Australia
Michael Weiss	Carleton University, Ottawa, Canada

TEAR 2008 Organizers

Pontus Johnson	Royal Institute of Technology, Stockholm, Sweden
Joachim Schelp	University of St. Gallen, Switzerland
Stephan Aier	University of St. Gallen, Switzerland

Table of Contents

I Fourth International Workshop on Engineering Service-Oriented Applications (WESOA 2008)

II Second International Workshop on Web APIs and Services Mashups (Mashups 2008)

III First International Workshop on Quality-of-Service Concerns in Service Oriented Architectures (QoSCSOA 2008)

IV Enabling Service Business Ecosystems (ESBE 2008)

V Third Workshop on Trends in Enterprise Architecture Research (TEAR 2008)

Fourth International Workshop on Engineering Service-Oriented Applications (WESOA 2008)

Introduction: Fourth International Workshop on Engineering Service-Oriented Applications (WESOA 2008)

Christian Zirpins[1], Guadalupe Ortiz[2], Yen-Jao Chung[3], and Wolfgang Emmerich[4]

[1] Universität Karlsruhe (TH), Germany
Christian.Zirpins@kit.edu
[2] University of Extremadura, Spain
GOBellot@unex.es
[3] IBM T.J. Watson Research Centre, United States
JYChung@us.ibm.com
[4] University College London, United Kingdom
W.Emmerich@cs.ucl.ac.uk

1 Workshop Goals and Contents

In large-scale software projects that increasingly adopt service-oriented software architecture and technologies, availability of sound systems engineering principles, methodology and tools for service-oriented applications is mission-critical for project success. However, engineering service-oriented applications poses specific requirements that differ from traditional software engineering and the discipline of software service engineering (SSE) is not yet established. Consequently, there is an urgent need for research community and industry practitioners to develop comprehensive engineering principles, methodologies and tool support for the entire software development lifecycle of service-oriented applications.

The WESOA series of workshops addresses challenges of software service engineering that arise from unique characteristics of service-oriented applications. Such applications are predominantly adopted in application domains that require complex as well as dynamic collaboration scenarios between autonomous individuals and thus have to represent many of their interaction patterns and organization principles. In many cases, service-oriented applications represent transactions of dynamic, process-driven business networks and drive interaction protocols between fluid configurations of autonomous service providers. In other cases, service-oriented applications are provided in the context of social communities, where they are created by a large number of members for very specific or even situational needs of long-tail economics. In such domains it is not enough to focus on complex distributed software systems alone but it is necessary to consider a broad socio-technical perspective.

It is the challenge of software service engineering to not only cope with these specific circumstances but to capitalise on them with radically new approaches. The WESOA series addresses these challenges and particularly concentrates on the aspects of service-oriented analysis and design that provide principles methodology and tool support to capture the characteristic requirements of service networks as well as service communities and transform them into reusable high-quality service system designs that underpin and drive the holistic service-oriented development lifecycle.

G. Feuerlicht and W. Lamersdorf (Eds.): ICSOC 2008, LNCS 5472, pp. 3–4, 2009.

For WESOA 2008 the international PC selected 9 out of 20 high-quality papers. These papers represent a rich variety of topics revolving around principles, methods and application domains of SSE with a special focus on analysis and design. A number of authors tackled fundamental SOA principles as regards *composite service design* (Feuerlicht), *unified management and governance* (Mueller et al.) as well as *SOA metrics* (Mamoun et al.). Other authors proposed specific SSE methods for *simulation* (Bartsch et al.), *automatic testing* (Kaschner et al.) and *behavioural analysis* (Aït-Bachir et al.) of SOA processes. This year, an increased number of contributions addressed software service design for specific application scenarios like *human-based e-services* (Schall et al.), *service value networks* (Bitsaki et al.) and *scientific workflow* (Tan et al.). The workshop was greatly enhanced by the keynote of Keith Duddy, in which he explored challenging gaps between the promises of modern service-oriented systems and the deficiencies of real world computing infrastructures. The workshop led to vigorous discussions around fundamental as well as specific characteristics of SOA and the challenges of establishing a new discipline of SSE.

2 Workshop Organisation

WESOA'08 was organised by an international group of researchers comprising the authors of this article. The event would not have been possible without the invaluable contribution of the international program committee. We would therefore like to thank the program committee members that include the following experts:

Sudhir Agarwal *(Univ. of Karlsruhe, DE)*, Marco Aiello *(Univ. of Groningen, NL)*, Sami Bhiri *(DERI Galway, IE)*, Jen-Yao Chung *(IBM Watson Research, US)*, Oscar Corcho *(Manchester Univ., GB)*, Vincenzo D'andrea *(Univ. Trento, IT)*, Valeria de Castro *(Univ. Rey Juan Carlos, ES)*, Gregorio Diaz *(Univ. of Castilla La Mancha, ES)*, Schahram Dustdar *(Tech. Univ. of Vienna, AT)*, Wolfgang Emmerich *(Univ. College London, GB)*, George Feuerlicht *(Sydney Univ. of Tech., AU)*, Stefan Fischer *(Univ. of Luebeck, DE)*, Howard Foster *(Imperial College London, GB)*, Paul Greenfield *(CSIRO, AU)*, Rannia Khalaf *(IBM Watson Research, US)*, Bernd Krämer *(Univ. of Hagen, DE)*, Winfried Lamersdorf *(Univ. of Hamburg, DE)*, Heiko Ludwig *(IBM Research, US)*, Tiziana Margaria-Steffen *(Univ. of Potsdam, DE)*, Michael Maximilien *(IBM Almaden Research, US)*, Massimo Mecella *(Univ. Roma LA SAPIENZA, IT)*, Harald Meyer *(HPI Potsdam, DE)*, Daniel Moldt *(Univ. of Hamburg, DE)*, Josef Noll *(Telenor, NO)*, Guadalupe Ortiz Bellot *(Univ. of Extremadura, ES)*, Rebecca Parsons *(ThoughtWorks, US)*, Greg Pavlik *(Oracle, US)*, Pierluigi Plebani *(Politecnico di Milano, IT)*, Franco Raimondi *(Univ. College London, GB)*, Wolfgang Reisig *(Humboldt-Univ. Berlin, DE)*, Thomas Risse *(L3S Research Center, DE)*, Norbert Ritter *(Univ. of Hamburg, DE)*, Stefan Tai *(Univ. of Karlsruhe, DE)*, Willem-Jan van den Heuvel *(Univ. of Tilburg, NL)*, Walid Gaaloul *(DERI Galway, IE)*, Jim Webber *(ThoughtWorks, AU)*, Christian Zirpins *(Univ. of Karlsruhe, DE).*

Finally, we would like to thank the ICSOC organisers, especially the workshop chairs George Feuerlicht and Winfried Lamersdorf as well as the local organisation chairs Helen Paik, Vladimir Tosic and Jian Yang for their guidance and support.

What Would Smart Services Look Like*

And How Can We Build Them on Dumb Infrastructure?

Keith Duddy

Queensland University of Technology / Smart Services CRC
126 Margaret St, Brisbane, QLD 4000, Australia

Abstract. The research for the next generation of service oriented systems development is well under way, and the shape of future robust and agile service delivery and service aggregation technologies is taking focus. However, the distributed computing infrastructure on which these systems must be built is suffering from years of "worse is better" thinking, and an almost invisible vendor fragmentation of the Web Services and Business Process Modelling spaces. The balkanisation of the basic technologies of service deployment and business process implementation threatens to undermine efforts to build the next generation of Smart Services. This paper is a summary of the keynote talk at WESOA 2008, which accounts for its informal style. It paints a picture of services futures, reveals the problems of the present tangle of technologies, and points to some practical initiatives to find the way out of the mess.

1 The Speaker, and His Journey "Up the Stack"

Keith Duddy is a graduate of the University of Queensland, and went from Honours in Computer Science to industry in 1990 where he worked on extending PC-Unix systems to use plug-in hardware which supported up to 32 serial ports for multiple terminal, printer and modem connections. His work included a mixture of hacking C code and supporting customer applications in 12 European countries. In 1993 he returned to the University of Queensland as a Research Assistant, and apart from a two year sojourn in the Health sector working for a government institute, he has been a professional researcher with Cooperative Research Centres – Australia's flagship program for industry, government and university collaboration.

Keith's research focus moved "up the stack" from serial communications protocols to working with CORBA Middleware and its supporting infrastructure services, including contributions to the CORBA Trader and CORBA Notifications standards. In 1996 Keith and a colleague organised for his employer, the Distributed Systems Technology Centre (DSTC), to be contracted by the Object Management Group (OMG) to develop and host *CORBAnet*, the CORBA

* The work reported in this paper has been funded in part by the Smart Services Co-operative Research Centre through the Australian Federal Government's CRC Programme (Department of Innovation, Industry, Science and Research).

G. Feuerlicht and W. Lamersdorf (Eds.): ICSOC 2008, LNCS 5472, pp. 5–14, 2009.

2.0 Interoperability showcase. This became an ongoing web-accessible demonstration of 12 separate CORBA implementations making calls on one another's distributed objects.

In the late nineties Keith became leader of the Pegamento project at the DSTC's successor, the Enterprise DSTC, investigating the confluence of middleware and workflow technologies, and began making extensive use of the DSTC's implementation of the MOF meta-modelling standard of OMG, which his DSTC colleagues had played a major role in specifying. Keith was the first author of a standard UML profile – for the representation of CORBA interfaces. As an OMG Architecture Board member, he was a co-author of the first technical white paper on the Model Driven Architecture, which was a concept already in use in EDSTC at the time.

His research focus continued to raise the level of abstraction in the specification and development of distributed systems – but with a focus on the flip side of that coin, reification of abstract specifications to real implementations by capturing parametric transformations from specification to implementations on multiple platforms. After being part of the large team that standardised the EDOC Metamodel and UML Profile in OMG, which resulted in some influential ideas being incorporated into UML 2.0, Keith was one of the authors and sponsors of the OMG's RFP for Model Transformation, to which his team contributed their in-house technology for MOF model transformations.

After EDSTC closed in 2004, Keith went to the Australian National e-Health Transition Authority (NEHTA) to apply his architectural and modelling experience in the health sector. This organisation has been applying Service Oriented Architecture principles, along with standards-based interoperability in an environment where documents and services need to be shared between a large number of public and private sector organisations, across three tiers of government, with diverse funding arrangements, and involving a large number of software vendors and developers. The experience of attempting to use COTS Web Services middleware which *claims* conformance to a variety of W3C and OASIS standards was eye-opening to say the least.

The latest phase of Keith's career is back in the CRC research framework – at the Smart Services CRC, where he is involved in two projects called *Service Aggregation* and *Service Delivery Framework*. These newly formed projects aim to bring the best of breed research in Service Oriented Computing at four Australian universities, together with applied research at SAP, Infosys and some government departments to offer a framework for practical SOA deployment.

2 What Is a Smart Service?

The short answer is that it is a marketing term to bring together various meanings of the term Service (economic, technical, political, business- and end-user-oriented) with an adjective to make it sound clever. The Smart Services CRC has an impressive portfolio of 11 projects with focus on finance, media and government which cover the spectrum of these meanings.

What we mean by *Smart Service* in the context of the Service Aggregation and Service Delivery Framework projects is easier to narrow down. We use Service Oriented Architectures as a basis for the meaning of *Service* – a function of an enterprise that is exposed through various technology-supported channels, and is amenable to re-use and composition into larger services which add value. Therefore a *Smart* Service is one that is better enabled for this purpose by approaches and technologies that are being developed in the CRC.

Smart Services, or the infrastructure in which they are deployed, will have some or all of the following properties.

2.1 Metadata

In order for a service client to be able to select the most appropriate service in a given environment for its needs and budget, the services available must be able to expose both their functional interface, as well as a set of non-functional properties as metadata. This includes concepts such as Quality of Service (QoS), which usually include things characterised as *-ilities*, like reliability, availability and execution time; as well as other properties like trust and reputation, operating hours, and other aspects of service provision usually found in Service Level Agreements (SLAs). In current service provision models, these are usually agreed on paper in a static framework of guarantees and compensations, and not exposed in a computer-readable form. Section 7 provides a suggested path forward for exposing service metadata.

2.2 Recursive Process Orientation

The emerging literature at ICSOC and other conferences and journals in the SOA space makes it clear that the name of the services game is to expose encapsulated business processes on one hand, and to create additional value by reusing services in the context of new business processes. The array of technologies used to choreograph and/or orchestrate service invocations is as wide as the Business Process Management field itself. The term *recursive* in this section header is important, however, as it requires the generalisation of the concepts of using services to enact business processes, as well as the creation of new services by exposing a wrapper interface to a business process itself. A process uses a service, which encapsulates a process, which invokes services, which encapsulate other processes, and so on.

The degree to which the process nature of services is exposed will vary, depending on factors such as traditional information hiding (which dictates opacity) and automated monitoring and use of formal methods for validation (which require increased transparency). To a large extent this will be dictated by organisational boundaries, with process exposed inside an enterprise, and hidden outside its borders to protect competitive advantage. A middle path being investigated by some researchers is one in which an abstract process definition is exposed, reflecting the logical execution of a service, but the complexity of the actual implementation is hidden.

Smart infrastructure to support service aggregation based on process definitions will be able to navigate the recursive process enacapsulations as deep as they are exposed within its context.

2.3 Service Selection and Substitution

When metadata is available to technical service execution frameworks, such as the increasingly dominant BPM paradigm, a service can be compared to others implementing the same logical function, using the same interface, and the one most fit for use can be selected. This is very simple when comparing only a single QoS parameter, however, various techniques exist to allow the optimisation of a whole process execution, involving multiple services, and taking into account multiple non-functional properties. These include Integer Programming, and Generic Algorithms, among many others. Some approaches apply local optimisation of the properties of each task in a process, and others consider the execution of a whole process and perform global optimisations.

However, the case where a large number of equivalent services with the same interface, varying only by QoS, are available is rare, and usually happens only inside a large "hub" or marketplace where the host defines the interfaces and makes them standards for provision of services into the market. Finding two Web Services "in the wild" with the same interface is exceedingly rare. This implies that some combination of the following approaches needs to be used:

1. Industry sectors and other groups using services will need to produce standardized "commodity" interfaces that will be implemented the same by all participants in that sector or grouping. (The political solution.)
2. Techniques need to be used which analyse services for compatibility, and transform similar interfaces to make them suitable for substitution for one another. (The technical solution.)

Although the precedent for industries in standardising data transfer formats, and sometimes interface definitions for their exchange, is there in bodies such as ISO, IEEE, ITU and even OMG, there is little evidence of this work being applied to things like standard WSDL definitions for domains. And for leading edge service integration activities in the current world, the need for Semantic-Web and AI-based techniques to discover service equivalences is obvious. Some of these technologies are well advanced in a research context, but mainstream efforts are still bound to manual wrappers around interfaces to bridge their syntactic differences, based on a human analysis of their semantic similarities. Once again, whether using ontologies and other semantic approaches, or AI techniques, the candidate services require good metadata and documentation above and beyond their interface definitions.

2.4 Constraint Enforcement

When using service aggregation techniques to bind multiple services into roles in a larger context, the ability for a high-level specification to indicate appropriate resource and other constraints is necessary, as the implementations of the

services are probably hidden from their BPM invokers. The classical example of such a constraint is the encapsulation of a legislative requirement for different entities to play the roles of *bank* and *auditor* in financial processes. Other example constraints include the selection of a service provider only from a list of nominated business partners, or the same storage provider to be used for all storage-related activities in a process for financial reasons. Smart infrastructure for service usage will extend the kinds of resource management functions of current Workflow environments to include the use of the metadata available about services (and their providers) to include constraint enforcement.

2.5 Metering and Billing

Ever wondered why there are only trivial examples of Web Services available on the open web? E.g. check the ISBN of a book, convert degrees Fahrenheit to degrees Celsius. It's probably because, without a standard per-use or per-volume metering and billing infrastructure available right alongside the WSDL endpoint information, no-one wants to make anything of value available when there's no person at the end of the interaction to view an advertisement – which is the standard entry-level charging model for the provision of high-value services to people through web browser interfaces.

To provide the incentive for service providers to offer their non-trivial services to a set of clients, the infrastructure needs to support a metering and billing apparatus. The current web economy is supported by consumers providing credit card details (or PayPal credentials) when they pay for services (usually on a subscription basis), or by advertising revenue when those services are presented through the web browser. The credit card/PayPal model typically applies when purchasing goods or services ordered, but not provided, via the web. There is no widespread model for charging for services used by aggregators to provide value-added services to users.

Truly smart services will have the ability to charge their users using multiple charging models, and unless a financial services company steps into this niche, other brokers will need to assume this role, and ultimately use the banking system to transfer funds at billing time. However, this is a space where a standard or *de facto* standard (analogous to a PayPal) for metering, billing and payment is needed.

3 Traders, Directories and Brokers

The idea of an *Open Service Marketplace* has been around since the early 1990s, and by 1995 ISO and the OMG had begun jointly standardising a "Trader" specification, which was one of the roles identified in the then recently published *ISO Reference Model for Open Distributed Processing*. RM-ODP (as it known for short) is a landmark publication which gave distributed systems researchers and practitioners the same kind of language for discussing distributed systems that the OSI specification had given network protocol designers a decade earlier. Both models continue to be relevant as reference sources today, even though networks still do not have seven layers, and the Trader role in distributed systems has failed to manifest.

Trader was a repository that contained *Service Offers* which had a *Service Type* describing both the interface and non-functional properties of a Service[1]. It was meant to allow service clients to specify queries across the properties which identified valid service offers for the needs of the client. Implementations of the OMG Trader, however, failed to find an application outside a few niche financial and telecoms management scenarios.

The next major hype about repositories of services came about with the release of the UDDI specification. The story that seemed to be believed by major vendors, including HP, IBM, Microsoft and SAP was that a global repository of Web Services would be available which would be populated by millions of endpoints, which could be navigated using their type and other metadata. The dream was backed up by a global scale implementation sponsored by IBM, Microsoft and SAP, and which lasted from 2001 until 2005, when the lights were turned off, due to lack of interest by Web Services developers.

So what's different about the concept of a Service Broker – a component that will store endpoints to multiple services and act as the place to discover the services needed in a particular marketplace? Sounds like déjà vu. Well, firstly, a broker will have a trusted relationship with both client and service provider (although it does not require that these parties are initially aware of one another), and a contract that permits it to charge for the services it delivers on behalf of the service providers, as well as taking a cut to sustain its own role in the marketplace.

Secondly, it will not only store the service endpoints for discovery by clients – it will be a logical intermediary in every interaction between clients and services. This mediation is mostly for the purposes of metering and billing. The broker knows which services are used, by whom, and for how long (or with what payload, or any other metering criteria that are useful), and has a contract with the clients which allows them to be billed for what they use.

4 How Does All This Relate to Web Services Standards?

The unstated assumption, since UDDI, is that when we speak of a "Web Service", that we are referring to a network-accessible endpoint which behaves in a predictable way (as specified in an interface description) when sent a message – much like its middleware predecessors, CORBA, DCE and Sun RPC. On the face of it, this appears be true, with the standards SOAP and WSDL being what makes this possible. But when one looks closer, it is clear that these standards offer so many variations in style (RPC, Document, Literal, Wrapped) that no vendor of Web Services toolkits offers all of the variations... and what's more, the intersection of the subset of the standards that they support is empty (or very small, depending on which vendor's products are under consideration).

[1] Interestingly, the Type Management function of RM-ODP which was intended to model these properties morphed into the OMG's Meta Object Facility (MOF), which emerged at the same time as the UML, and has since become intervolved with that standard.

A whole industry consortium was formed on the basis that the standards from W3C and OASIS offer no profiles for their usage which can foster interoperability: Web Services Interoperability (WSI). This body attempted to overcome the ambiguities and range of implementation choices offered by the standards, but the evidence seems to be that they have failed to do so, with even the WSI-Basic profile for the use of Web Services offering three variations, and some of the major toolkit vendors (like Microsoft) opting out of supporting even one of these. Web pages like [7] and [8] give an indication of proliferation of N by N interoperability problems that programmers face when using more than one Web Services toolkit. An example of an attempt to specify a common profile that makes no specific product references can be seen in [1], specified by NEHTA. The problem with this sort of profile is that its implementation in a particular toolkit often requires hand modification of SOAP messages after they have been created in a non-conformant manner by the generated code, and no documentation exists in the toolkit to assist.

The current facts of the situation are that out-of-the box interoperability, without hacking the XML in the SOAP messages for a service by hand, does not exist between the two most popular WS frameworks: JAXWS and .Net. And this is just using the most common WS-* standards: WSDL, WS-Addressing and WS-Security. There are more than 50 current standardisation efforts in W3C and OASIS that attempt to augment these basic components to do every other middleware function imaginable, from Transaction and Policy, to Reliable Messaging, to BPM. The dependency matrix between the standards is highly complex, and often simply contradictory, with many of the standards gratuitously overlapping in functionality. The principle of separating interface from implementation is paid lip service, but in reality the only way to get interoperability is to implement all services and their clients using the same vendor's platform.

The current .Net toolkit does not even allow the use of a WSDL with arbitrary XML payloads – even when using the right combination of literal and wrapped conventions and the right choice of security signing before encrypting, rather than the other way around. The Microsoft response to support calls to assist in delivering HL7 (a heath domain standard) XML payloads using Web Services is a recommendation to design everything from scratch in C# using Visual Studio, and then generate the WSDL, as "WSDL-first development is not supported".

5 What about the Web Services Success Stories?

A number of web-based platforms and Software-as-a-Service offerings are usually held up by Web Services spruikers as *WS Success Stories*. These include Rearden Commerce, which is the platform used in American Express's Axiom corporate travel spend-control site; RightNow.com and SalesForce.com, which are CRM systems hosted on behalf of clients in larger and smaller companies respectively; and last, but not least, Amazon.com, which integrates thousands of retailers' product offerings through its portal using WS-* specifications. So when these successful businesses base their multi-billion dollar strategies on WS, how can I claim that it's a big non-interoperable mess?

What is really happening is that WS-* is hidden from the vast majority of users of these systems - who mainly access their services using POWP (plain old web pages). The WS part of these companies' solutions is very carefully managed between the hub, and their strategic partners – a so-called Walled Garden. The hub controls all of the WSDL defined, and they use only the simplest WSDL definitions. These interfaces convey only very simple XML payloads (in the Amazon.com case, everything in the WSDL is a string element). Virtually none of the other WS-* standards outside of WSDL and SOAP are used (the exception being some use of WS-Addressing). WS is used to connect to a proprietary set of functionality to support user profiles, events and notifications, and payment and charging infrastructure.

Another interesting thing to note is that Google has decided that WS-* is not ready for prime time, and instead they support a large set of language library APIs that do proprietary communications back to the Googleplex.

6 What Other Challenges Do We Face?

6.1 Business Process Management

One of the key emerging technologies for aggregating services into useful business functions is Business Process Management (BPM), which is being used increasingly to choreograph services as well as the activities of people. Unlike the service invocation space, in which WSDL has emerged as the *lingua franca* of interface definitions (despite the support for the standard being partial and often non-overlapping in the various implementations), BPM is still in a state where a number of competing languages are being used to define processes.

BPMN is favoured by analysts, and has increasing tool support, although the standard does not have a well defined execution semantics. UML Activity Graphs have a petri-net inspired token-passing semantics (with "semantic variations"), and is supported as a graphical language by most UML tools, although most of these do not perform the necessary well-formedness checks, and thus leave most Activity Graphs that I have ever seen with errors which make them essentially semantic-free. BPEL is based on Pi-calculus, and has no formal mappings to the graphical languages, and no graphical syntax of its own. YAWL is formally defined based on modified petri-nets, but has only a single open-source implementation.

There are myriad other languages for BPM, all of which are fundamentally incompatible, and therefore there does not exist any general way of translating between them, as can be done with RPC interface descriptions of many kinds. This is a field in which it is also unlikely that the industry will settle on a single formalism, as there are constant innovations (of the reinventing the wheel sort), which throw in new concepts and notations (as well as a host of partially defined, or informal, description techniques) to muddy the waters.

6.2 Quality of Service

The term QoS has been refined and formalised in the context of networking and telecoms management over the last decade or so, and companies like Cisco have

made a fine art of managing basic networking services. However when applying the basic idea – of dynamically available meta-data about the performance of the non-functional aspects of a service – to general computing services, a lot of confusion exists in the literature, and some basic questions are unanswered:

- Is *execution time* end-to-end, or just at the server. Is it average, or maximum, and who measures it?
- Is *price* the access price (e.g. searching the catalogue) or the price of the eventual commodity (e.g. buying an MP3). Is price even a QoS?
- Who rates *reputation and trust*? Who stores it? Is the mechanism revealed (number of reviews, etc)?
- Who stores any QoS property? And in what format?

Most academics working with optimising service aggregations make assumptions that QoS is available and accurate, that it is variable and differentiated between otherwise identical services. Whereas the large deployments of WS that we know of use mechanisms like Service Level Agreements to ensure that alternative providers of services all comply to the same level of quality, and none of them store and/or calculate, or query, QoS properties for individual services at runtime.

7 Some Practical Initiatives to Overcome Challenges

7.1 Service Description Meta-models

The PhD work of Justin O'Sullivan [4] is fully documented at http://service-description.com/. It resolves many of the questions asked about QoS, and has been used in European Union projects, and is included in new product development underway at SAP.

7.2 Domain Specific Languages

DSLs raise the level of abstraction when specifying the operation of software (and hardware) in a particular domain setting. Many industries have successfully standardised documents and processes that apply across a sector, using MOF, XML, UML and other abstractions. Much of the promise of OMG's Model Driven Architecture is embodied in the use of DSLs to abstract away from much of the technology and implementation detail when specifying a system. Even when no automatic code generation is possible, the effort of extracting irrelevant detail from a problem specification makes the design more understandable, more portable, and more amenable to automated code generation or bespoke platform construction techniques (such as software factories).

7.3 KISS Initiative

The KISS (Knowledge Industry Survival Strategy) Initiative puts forward a Modeling Tool Interoperability Manifesto and Projects. In addition, the initiative in running a Workshop Series at major conferences throughout 2009/2010. See http://www.industrialized-software.org/kiss-initiative.

8 Conclusion

Standards development in the Web Services space has hit an all-time quality low, with conformance being left unconsidered, or at best patched in after the fact by other standards attempts. Architectural oversight is almost non-existent across the fifty plus ongoing WS-* standards processes in W3C and OASIS, and many potential standards overlap in functionality and intent, with support from vendors very fragmented.

Thankfully in another decade it is very likely that we will be throwing this all away and starting over for a third generation of middleware standards, according to the perceived needs and fashions of the time.

Smart Services will be the combination of a range of techniques to describe, coordinate, allow invocations of, and allow billing of individual Web Services. These services are likely to be kept to a small common subset of interface types and styles, and use simple XML types to convey payloads.

The discovery, substitution and construction of variations of services will also be facilitated by Smart Services infrastructure. This will contain the "duct-tape and ticky-tacky" needed to glue together the incompatible parts of the WS world.

The aggregation of simple (and wrapped complex) services will increasingly use BPM techniques, but this field will necessarily be fragmented according to the kind of BPM language used, as there is no logical path to translating BPM models between the different formalisms.

References

1. NEHTA: Web Services Profile, Version 3.0 (December 1, 2008)
2. Hamadi, R., Benetallah, B.: A Petri Net-based Model for Web Service Composition. In: Proceedings of Fourth Australasian Database Conference (2003)
3. Zeng, L., Benetallah, B., Ngu, A.H.H., Dumas, M., Kalaggnanam, J., Chang, H.: QoS-Aware Middleware for Web Services Composition. IEEE Transactions on Software Engineering 30(5) (May 2004)
4. O'Sullivan, J., Edmond, D., ter Hofstede, A.: What's in a Service? Distributed and Parallel Databases 12(2-3), 117–133 (2002)
5. Recker, J., Mendling, J.: On the Translation between BPMN and BPEL: Conceptual Mismatch between Process Modeling Languages, http://citeseer.ist.psu.edu/recker06translation.html
6. Cohen, F.: Understanding Web service interoperability: Issues in integrating multiple vendor Web services implementations (Februray 2002), http://www.ibm.com/developerworks/webservices/library/ws-inter.html
7. Simon Guest: Top Ten Tips for Web Services Interoperability (August 2004), http://blogs.msdn.com/smguest/archive/2004/08/12/213659.aspx
8. Ye, W.: Web Services Interoperability between J2EE and .NET (April 2005), http://www.developerfusion.com/article/4694/web-services-interoperability-between-j2ee-and-net-part-1/

Design of Composable Services

George Feuerlicht[1,2]

[1] Department of Information Technology,
University of Economics, Prague, W. Churchill Sq. 4, Prague, Czech Republic
[2] Faculty of Engineering and Information Technology,
University of Technology, Sydney
P.O. Box 123 Broadway, Sydney, NSW 2007, Australia
jiri@it.uts.edu.au

Abstract. Service composition methods range from industry standard approaches based on Web Services and BPEL to Semantic Web approaches that rely on AI techniques to automate service discovery and composition. Service composition research mostly focuses on the dynamic (workflow) aspects of compositions. In this paper we consider the static component of service composition and discuss the importance of compatibility of service interfaces in ensuring the composability of services. Using a flight booking scenario example we show that reducing the granularity of services by decomposition into service operations with normalized interfaces produces compatible interfaces that facilitate service assembly. We then show how relational algebra can be used to represent service operations and provide a framework for service assembly.

Keywords: Service composition, service assembly, service reuse.

1 Introduction

In general, the specification of service compositions consists of two parts: the static part that involves the definition of services, including the service operations and their interfaces, and the dynamic part that defines the associated process workflow. It can be advantageous to treat the design of the static part of service compositions separately, as the service operations can then be reused in the context of various process specifications [1]. We refer to this static part of service composition as service assembly in this paper. In this context, the design of the inbound and outbound message structures is of paramount importance as it determines the compatibility of service interfaces, and consequently the composability of services into higher level business functions. A key determinant of service composability and reuse is service granularity, i.e. the scope of functionality that individual services implement. Increasing the scope of the functionality implemented by a given service reduces the potential for reuse and therefore makes the assembly of services more problematic. In previous publications we have described a methodological framework for the design of services based on the data properties of interface parameters that aims at achieving optimal level of service granularity [2, 3]. In this paper we extend this framework and consider the problem of service assembly. Unlike some authors, for example [4-8] who consider service composition a run-time problem and apply semantic

G. Feuerlicht and W. Lamersdorf (Eds.): ICSOC 2008, LNCS 5472, pp. 15–27, 2009.

techniques to run-time resolution of compatibility conflicts, we regard service assembly a design-time concern and focus on specifying compatible services interfaces. Furthermore, our analysis assumes that we are dealing with domain-wide services based on an industry standard specification, avoiding incompatibilities arising from services designed by individual service providers. Service assembly is a recursive process that produces high-level (coarse-grained) services as compositions of elementary (fine-grained) services. A key objective of service design is to ensure that services are both composable and reusable so that higher level services can be implemented as assemblies of mutually independent elementary services. Service compositions are typically implemented using languages such as BPEL (Business Process Execution Language) to form complete business processes, and we adopt the BPEL composition model as the basis for our analysis [9]. BPEL compositions involve implementing higher level business functions using previously defined Web Services accessible via partner links and externalizing the resulting functionality of the composite service via a WSDL interface. This process can be repeated recursively, so that complex high-level business functions can be implemented by aggregation of lower level services [10]. The BPEL model is a message-based paradigm and the communication between Web Services involves mapping the results of service invocations between the outbound and inbound messages of service interfaces (i.e. the signatures of Web Service operations). Local and global BPEL variables are used to store and manipulate the intermediate results of service invocations. It follows that composability of services is dependent on the compatibility of interfaces of service operations involved in the composition.

We have argued elsewhere that service interfaces can be treated as data parameters, and that data engineering principles apply to the design of services [2]. In this paper we extend this work to considerations of service composability. More specifically, we argue that composability of services depends on compatibility of service interfaces within the service assembly, and show that relational algebra formalism can be used to represent the static part of service compositions. In the next section (sections 2) we explore service composability in more detail and discuss the relationship between service reuse and composability. We then illustrate the process of decomposing coarse-grained services into elementary (fine-grained) services with normalized interfaces that facilitate service composition using a flight booking scenario (section 3). In the next section (section 4) we show how relational algebra formalism can be used to represent service operations and to describe service assembly. In the final section (section 5) we summarize the main contributions of this paper, discuss related work and outline further research.

2 Considerations of Service Reuse and Composability

Composability of services is closely related to service reuse. Many experts believe that reuse is inherent to SOA (Service-Oriented Architecture), and studies show that organizations regard reuse as the top driver for SOA adoption [11]. However, in practice reuse can be difficult to achieve and involves design-time effort to identify and design reusable services. Once services are published it becomes very difficult to improve the level of service reuse by runtime intervention, or by modifying the externalized service interfaces. It can be argued that the perception of improved reuse can be mainly attributed to the ability to derive business value from legacy applications by

externalizing existing functionality as Web Services [12]. Others have noted that the relatively low levels of service reuse can be attributed to poor design [13]. For the purposes of this analysis it is important to understand the mechanism for service reuse and how it differs from earlier approaches to software development. The mechanism for service reuse is service aggregation and we can define service reuse as the ability of a service to participate in multiple service assemblies/compositions; in this sense, service composability can be regarded as a measure of service reuse. Service composition implements complex application functionality (for example, a composite service can implement a hotel reservation, airline booking and a car rental to form a complete travel agency service) by means of recursively composing services [14]. In order to facilitate composition, the constituent services need to have characteristics which allow reuse in composing services.

Services share the basic characteristics of components such as modularization, abstraction and information hiding, and design strategies used in earlier software development approaches can be adapted to service design. However, there are significant challenges to overcome as services are typically implemented at a higher level of abstraction than components, and reuse potential is limited by the extensive use of coarse-grained, document-centric services. Another factor that makes achieving good levels of service reuse particularly challenging is that service-oriented applications tend to span organizational boundaries and the design of services frequently involves industry domain considerations. Emerging vertical domain standards such as the Open Travel Alliance (OTA) [15] specification of (XML) message formats for the travel industry domain are typically developed by committees or consortia which tend to operate on a consensus basis and pay little attention to software design. Although industry-wide standards are an essential prerequisite for e-business interoperability, standardization of message structures and business processes alone does not ensure reusability of services. The resulting message structures typically include a large number of optional elements and embedded instructions that control the processing of the documents [16]. This results in excessively complex and redundant data structures (i.e. overlapping message schemas) that include a large number of optional data elements introducing high levels of data and control coupling between service operations [17]. Consequently, Web Services based on such message formats do not exhibit good levels of reuse and composability [18]. It is often argued that standardization leads to reuse [19], but in order to achieve high levels of service reuse and composability, detailed consideration needs to be given to service properties at design-time. More specifically, services should be self-contained, have clearly defined interfaces that are compatible across the domain of interest. These requirements lead to a consideration of cohesion (i.e. maximization of service cohesion) and coupling (i.e. minimization of coupling between services) resulting in fine-grained services that are associated with improved level of service composability [20].

3 Identifying Composable Services

Most vertical-domain applications are characterized by coarse-grained services that typically encapsulate high-level business processes and rely on the exchange of composite XML documents to accomplish business transactions. This mode of operation is widely adopted by the SOA practitioners for developing Web Services applications

to improve performance and reduce the number of messages that need to transmitted to implement a specific business function [21]. Consider, for example, travel Web Services based on the OTA specification implement flight booking business process for a specific itinerary using two request/response message pairs: OTA_AirAvailRQ/ OTA_AirAvailRS and OTA_AirBookRQ/OTA_AirBookRS. The OTA_AirAvailRQ/ OTA_AirAvailRS message pair includes the data elements of requests and responses for airline flight availability and point of sale information [15]. This situation is illustrated in Figure 1. The Availability Request message requests flight availability for a city pair on a specific date for a specific number and type of passengers, and can be narrowed to request availability for a specific airline, flight or booking class on a flight, for a specific date. The Availability Response message contains the corresponding flight availability information for a city pair on a specific date. The Availability request/response interaction is (optionally) followed by the Booking request/response message exchange. The Book Request message is a request to book a specific itinerary for one or more passengers. The message contains origin and destination city, departure date, flight number, passenger information, optional pricing information that allows the booking class availability and pricing to be re-checked as part of the booking process. If the booking is successful, the Book Response message contains the itinerary, passenger and pricing information sent in the original request, along with a booking reference number and ticketing information.

The use of such complex (coarse-grained) data structures as payloads of Web Services SOAP message improves performance, but significantly reduces reuse potential [22].

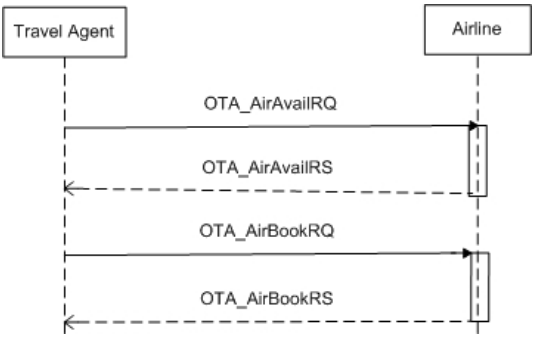


Fig. 1. OTA flight availability and booking message sequence

Decomposition of coarse-grained services into fine-grained (elementary) service operations improves the opportunity for reuse and is a necessary precondition for composability of services. In order to facilitate service composition, detailed consideration needs to be given to service interfaces at design-time to ensure that individual services are mutually compatible across a collection of related services. This leads to the requirement for matching of interface parameters, so that for example, the BookingReference (output) parameter of the BookFlight operation matches the BookingReference (input) parameter of the MakePayment operation. We explore this aspect of service composition in the following sections, using an airline flight booking scenario introduced in this section.

3.1 Decomposition of the Travel Booking Service

We make a number of simplifying assumptions including that the flights are one-way with no stopovers and that flights for a given FlightNumber depart every day of the week at the same time (DepartureTime). These simplifications make the example easier to follow while maintaining good correspondence to the *real-world* situation. Unlike the coarse-grained message interchange pattern used by the OTA specification (illustrated in Figure 1) this scenario breaks down the business function into fine-grained service operations that closely match the requirements of the flight booking dialogue. As argued elsewhere [13, 23], the benefits of this fine-grained design include improved cohesion, reduction in coupling and better clarity. But, also importantly, reducing granularity leads to improved flexibility, reusability and composability of services, so that for example, the payment operation (MakePayment) is now separate from the booking operation (BookFlight). As a result, it is possible to hold the booking without a payment, furthermore, the MakePayment operation can be reused in a different context, e.g. in a hotel booking service. In order to identify candidate service operations, we first model the flight booking dialogue using a sequence diagram (Figure 2), and then define the corresponding service interfaces using simplified OTA data elements. Similar to the OTA message sequence shown in Figure 1, the sequence diagram in Figure 2 describes the interaction between a travel agent and an airline. Each message pair consists of a request (RQ) message and a response (RS) message that together form the interface of the corresponding candidate service operation. We can now describe the flight booking function in more detail using a composition of 4 service operations: FlightsSchedule, CheckAvailability, BookFlights, and MakePayment as identified in the sequence diagram in Figure 2. The flight booking dialogue proceeds as shown in Figure 3. The traveler supplies the values for DepartureCity, DestinationCity, and DepartureDate as input parameters for the FlightsSchedule operation. The

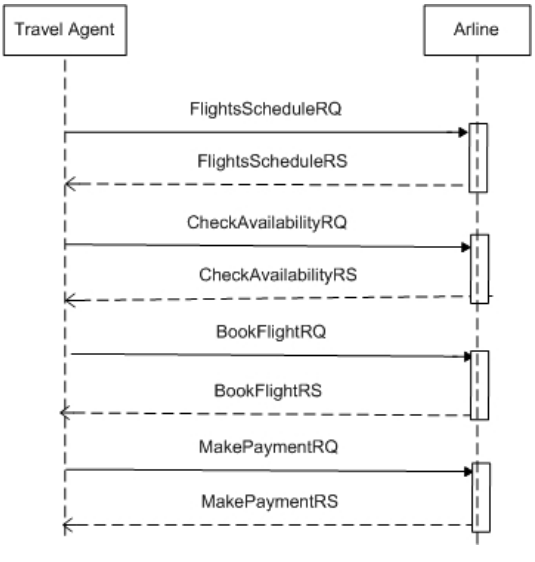


Fig. 2. Modified flight availability and booking message sequence

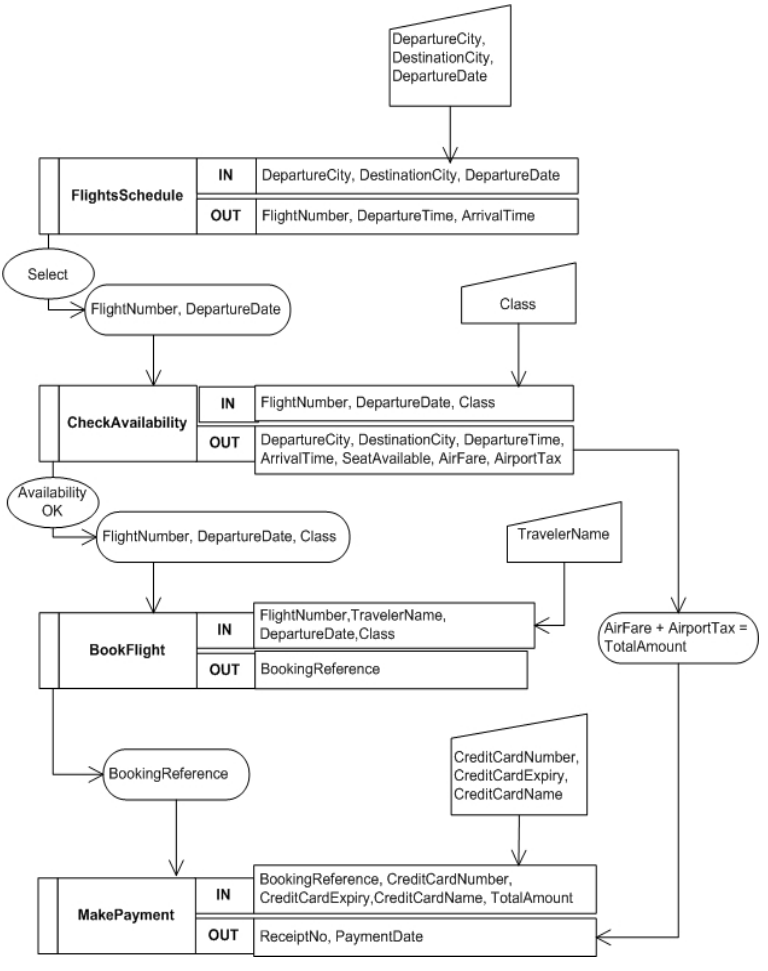


Fig. 3. Flight booking dialogue using fine granularity services

output of the FlightsSchedule operation produces a list of scheduled flights, i.e. corresponding values of FlightNumber, DepartureTime, and ArrivalTime. The traveler then selects a suitable flight (i.e. FlightNumber and DepartureDate) supplies the value of Class (e.g. economy); the values of FlightNumber, DepartureDate and Class then form the input for the CheckAvailability operation.

The output of the CheckAvailability operation includes information about flight availability (SeatAvailable) and pricing information (Airfare and AirportTax).

Assuming that seats are available for the selected flight the traveler proceeds to book the flight using the BookFlight operation that takes the values of FlightNumber, DepartureDate, Class, and TravelerName as the input, and produces BookingReference as the output. Finally, the traveler makes a payment using the MakePayment operation supplying, credit card information (CreditCardNumber, CreditCardExpiry, CreditCardName). The MakePayment operation accepts the input parameters BookingReference and TotalAmount (sum of Airfare and AirportTax) generated by the

BookFlight and SelectFlight operations, respectively, and produces ReceiptNo and PaymentDate as the output parameters.

Table 1 describes the service interfaces for the operations FlightsSchedule, CheckAvailability, BookFlights, and MakePayment showing the input and output parameters.

Table 1. Flight availability and booking service operations

Operation	Input Parameters	Output Parameters
FlightsSchedule	DepartureCity, DestinationCity, DepartureDate	FlightNumber, DepartureTime, ArrivalTime
CheckAvailability	FlightNumber, DepartureDate, Class	DepartureCity, DestinationCity, DepartureTime, ArrivalTime, SeatAvailable, AirFare, AirportTax
BookFlight	FlightNumber,TravelerName, DepartureDate,Class	BookingReference
MakePayment	BookingReference, CreditCardNumber, CreditCardExpiry,CreditCardName, TotalAmount	ReceiptNo, PaymentDate

3.2 Data Analysis of Service Interfaces

Although the data used by the flight booking scenario is typically stored in different databases belonging to different participants in the business process (i.e. travel agent, airline, etc.), for the purposes of this analysis we assume that this data can be described by a common (global) database schema. Although not explicitly defined, this common schema is implicit in the industry-wide message specifications (i.e. OTA message schema specification, in this instance). We note here that we do not make any assumptions about how and where the data is stored; we simply use the underlying data structures to reason about the composability of services. We also do not consider issues related to state maintenance, as these are orthogonal to the considerations of service composability. OTA specification also assumes that the data transmitted in XML messages is stored persistently in the target databases and provides a number of messages to synchronize the data across the various participants (e.g. OTA_ UpdateRQ/RS, OTA_DeleteRQ, etc.).

We can now proceed to analyze the underlying data structures as represented by the data elements in the interfaces of the service operations. Data analysis of the content of the interfaces of service operations in Table 1 produces a set of 5 normalized relations that constitute the database schema associated with the flight booking business function:

Flights (FlightNumber,DepartureCity,DestinationCity, DepartureTime,ArrivalTime)

Schedule (FlightNumber,DepartureDate,AircraftType)

Availability (FlightNumber,DepartureDate,Class,SeatAvailable,AirFare,AirportTax)

Bookings (BookingReference,TravelerName,FlightNumber, DepartureDate,Class,Seat)

Payments (ReceiptNo,PaymentDate,CreditCardNumber,CreditCardExpiry,CreditCardName, BookingReference)

Given the above normalized relations, we can observe by inspecting Figure 3 that the assembly of the flight booking service takes place by passing the values of the key attributes between the service operations. For example, the composite key of the Availability relation (FlightNumber, DepartureDate, Class) forms the data flow between CheckAvailability and BookFlight operations, and the BookingReference (i.e. the primary key of the Bookings relation) constitutes the data flow between BookFlight and MakePayment operations. This indicates that data coupling between the service operations is minimized as the elimination of any of the parameters would inhibit composition, e.g. removing Class from the dataflow between CheckAvailability and BookFlight operations would prevent the composition of the flight booking business function. Furthermore, the interface parameters are mutually compatible as they share common data parameters. In summary, it can be argued that the normalization of service interfaces results in service operations with high levels of cohesion, low levels of coupling and mutually compatible interfaces; properties that significantly improve service reusability and composability.

4 Describing Service Assembly Using Relational Algebra Operations

In the previous section we have described the process of decomposition of services into elementary service operation; service assembly involves reversing this process and combining services based on interfaces data parameters. We have noted that coupling between service operations involves data parameters that correspond to the keys of the underlying relations. We can use this observation to express services using relational algebra expressions or operator trees over the underlying schema [24]. For example, the FlightSchedule operation can be expressed as:

$$\mathbf{PJ}_{\text{FlightNumber,DepartureDate,ArrivalTime}}\ \mathbf{SL}_{\text{P1}}\mathbf{JN}_{\text{FlightNumber=FlightNumber}}(\text{Schedule, Flights}),$$

where PJ, SL, and JN represent projection, selection, and join operations respectively, and P1 is a selection predicate (e.g. DepartureCity = “Sydney“ and DestinationCity = “Melbourne“ and DepartureDate= “31-May-2007“).

We can now express the operation FlightSchedule and CheckAvailability in relational algebra, for clarity substituting values into the predicates using selection specification as shown below:

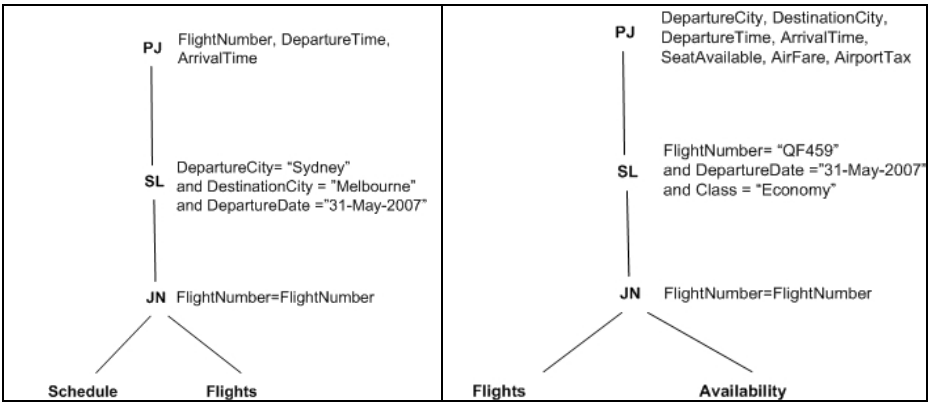
FlightSchedule:

$$\mathbf{PJ}_{\text{FlightNumber,DepartureDate,ArrivalTime}}\mathbf{SL}_{\text{DepartureCity=“Sydney“ and DestinationCity=“Melbourne“ and DepartureDate=“31-May-2007“}}\mathbf{JN}_{\text{FlightNumber=FlightNumber}}(\text{Schedule,Flights})$$

CheckAvailability:

$$\mathbf{PJ}_{\text{DepartureCity,DestinationCity,DepatureTime,ArrivalTime,SeatAvailable,Airfare,AirportTax}}$$
$$\mathbf{SL}_{\text{FighNumber=“QF459“ and DepartureDate=“31-May-2007“ and Class=“Economy“}}$$
$$\mathbf{JN}_{\text{FlightNumber=FlightNumber}}(\text{Flights,Availability})$$

Alternatively, the operations FlightSchedule and CheckAvailability can be expressed in the form of operator trees as shown in Figure 4. Figure 4(a) shows the operator tree for the FlightSchedule operation. The output parameters of the FlightSchedule operation (FlightNumber, DepartureDate, Arrival-Time) appear at the top of the operator tree, and the input parameters (DepartureCity="Sydney" and DestinationCity="Melbourne" and DepartureDate ="31-May-2007") form the predicate of the SL (select) operation. Now, assuming that the traveler selects FlightNumber = "QF459", DepartureDate ="31-May-2007" and Class = "Economy", this triplet of values forms the input for the CheckAvailability operation shown in Figure 4(b).



(a) (b)

Fig. 4. Operator tree representing the FlightSchedule (a) and CheckAvailability operation (b)

The output parameters of the CheckAvailability operation (DepartureCity, DestinationCity, DepartureTime, ArrivalTime, SeatAvailable, AirFare, AirportTax) appear at the top of the operator tree in Figure 4(b).

Having expressed service operations using relational algebra formalism we can now proceed and express service assemblies as an algebraic expression [24]. So that, for example we can combine the operations CheckAvailability and FlightSchedule to produce a composite AirAvailability operation:

AirAvailability:PJ$_{\text{DepartureCity,DestinationCity,DepatureTime,ArrivalTime,SeatAvailable,Airfare,AirportTax}}$
SL$_{\text{FighNumber="QF459" and DepartureDate="31-May-2007" and Class="Economy"}}$
JN$_{\text{FlightNumber=FlightNumber}}$(Flights,Availability,Schedule)

The expression for the AirAvailability operation uses the equivalence: R:$_{P1}$ **JN**$_{F}$ S:$_{P2}$ = R **JN** S: $_{P1 \text{ AND } P2 \text{ AND } F}$, where R and S are relations, P1 and P2 are selection predicates, and F is the join expression. Figure 5 show the resulting AirAvailability operation expressed as an operator tree. We now show the composite operation AirAvailability in the usual form that includes input and output data parameters, and can be mapped into a WSDL specification:

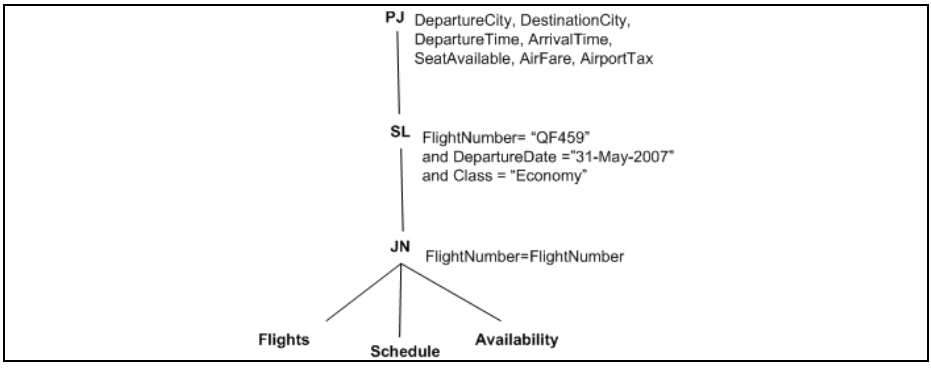


Fig. 5. Operator tree of a composite operation AirAvailability

AirAvailability:
(IN: FlightNumber, DepartureDate, Class,
OUT: DepartureCity, DestinationCity, DepatureTime, ArrivalTime,
SeatAvailable, Airfare, AirportTax)

Using this approach provides a formal framework for static service composition that allows decisions about the level of service aggregation to be based on considerations of tradeoffs between complexity of run-time dialogue (i.e. *chattiness* of services) on one hand, and software engineering properties of services such reusability, on the other hand. The designer may, for example, decide to implement the CheckAvailability and FlightSchedule service operations *internally* (i.e. within the service provider system) and externalize the composite operation AirAvailability, gaining the benefits of reuse (and composability) for internal applications, and at the same time reducing the number of messages needed to implement the flight booking dialogue. This solution is similar to the *remote façade* design pattern use to implement coarse-grained interfaces in object-oriented applications [25].

4 Conclusions and Related Work

Service composition methods range from industry standard approaches based on Web Services and BPEL [26] that focus on defining the workflow of Web Services execution, to Semantic Web approaches that employ AI techniques to automate service discovery and composition [27-28]. Service composition can be regarded as a special category of the software composition problem that has been investigated in the context of object-oriented software [29] and in the general area of software composition [30]. Many researchers have applied formal methods and developed specialized composition languages to address the problem of composition [31-32]. As noted in the introduction, service composition research mostly focuses on the dynamic (workflow) aspects of compositions. We have argued in this paper that from the viewpoint of service reuse and composability, the static part that involves the definition service operations and their interfaces is of key importance. The design of the inbound and outbound message structures determines the compatibility of service interfaces, and

consequently the composability of services into higher level business functions. The main contribution of this paper is to show that composability (and reuse) of services can be facilitated by designing services with compatible service interfaces and that service assembly can then be achieved by service aggregation over the key attributes of the underlying schema. We have also shown that relational algebra formalism can be applied to the problem of representing service operations, and defining service assemblies. Service decomposition and assembly framework based on data normalization and relational algebra operations can provide a theoretical framework for combining service operations to achieve desired business functionality and at the same time maintaining high levels of service reuse.

A number of aspects of this approach deserve further investigation. Firstly, the potential of using algebraic equivalence transformations for identifying alternative composition strategies and for optimizing the level of service granularity needs further study [24]. Another potential use of the relational algebra formalism is in the area of verification of the correctness of compositions, i.e. using algebra to prove the correctness of static compositions. Finally, the examples used in the previous section (section 3) involve services that represent *query* operations, i.e. operations that return data values given a set of input parameters. Service operations that result in state change, i.e. functions that generate new data values (e.g. Bookings and Payments) cannot be directly represented by algebraic expressions and require further analysis to enable their incorporation into this framework.

Acknowledgements

We acknowledge the support of MŠMT ČR in the context of grant GAČR 201-06-0175 “Modification of the model for information management”.

References

1. Thöne, S., Depke, R., Engels, G.: Process-oriented, flexible composition of web services with UML. In: Olivé, À., Yoshikawa, M., Yu, E.S.K. (eds.) ER 2003. LNCS, vol. 2784, pp. 390–401. Springer, Heidelberg (2003)
2. Feuerlicht, G.: Design of Service Interfaces for e-Business Applications using Data Normalization Techniques. Journal of Information Systems and e-Business Management, 1–14 (2005) ISSN 1617-98
3. Feuerlicht, G., Meesathit, S.: Design framework for interoperable service interfaces. In: The Proceedings of 2nd International Conference on Service Oriented Computing, New York, NY, USA, November 15 - 19, 2004, pp. 299–307 (2004) ISBN 1-58113-871-7
4. Wen-Li Dong, H.Y., Zhang, Y.-B.: Testing BPEL-based Web Service Composition Using High-level Petri Nets. In: 10th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2006), Hong Kong, pp. 441–444 (2006)
5. San-Yih Hwang, E.-P.L., Lee, C.-H., Chen, C.-H.: On Composing a Reliable Composite Web Service: A Study of Dynamic Web Service Selection. In: IEEE International Conference on Web Services (ICWS 2007), pp. 184–191 (2007)

6. Keita, F., Tatsuya, S.: Dynamic service composition using santic information. In: Proceedings of the 2nd international conference on Service oriented computing. ACM, New York (2004)
7. Freddy, L., et al.: Towards the composition of stateful and independent semantic web services. In: Proceedings of the 2008 ACM symposium on Applied computing. ACM, Fortaleza (2008)
8. Meng, X., et al.: A Dynamic Semantic Association-Based Web Service Composition Method. In: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence. IEEE Computer Society, Los Alamitos (2006)
9. Arkin, A., et al.: Web Services Business Process Execution Language (WS-BPEL). OASIS 2, Version, http://www.oasis.org
10. Yang, J.: Service-oriented computing: Web service componentization. Communications of the ACM 46(10), 35–40 (2003)
11. Hurwitz, J., Bloor, R., Baroudi, C.: Thinking from Reuse - SOA for Renewable Business (2006) (cited December 13, 2007), http://www.hurwitz.com/PDFs/IBMThinkingfromReuse.pdf
12. Feuerlicht, G., Wijayaweera, A.: Determinants of Service Resuability. In: The Proceedings of 6th International Conference on Software Methodologies, Tools and Techniques, SoMet 2006, Rome, Italy, November 7-9 (2007) ISBN 0922-6389
13. Sillitti, A., Vernazza, T., Succi, G.: Service oriented programming: A new paradigm of software reuse. In: Gacek, C. (ed.) ICSR 2002. LNCS, vol. 2319, pp. 268–280. Springer, Heidelberg (2002)
14. Dustdar, S., Schreiner, W.A.: A survey on web services composition. International Journal of Web and Grid Services 1(1), 1–30 (2005)
15. OTA, OTA Specifications (2008) (cited May 6, 2008), http://www.opentravel.org/Specifications/Default.aspx
16. Feuerlicht, G.: Implementing Service Interfaces for e-Business Applications. In: The Proceedings of Second Workshop on e-Business (WeB 2003), Seattle, USA (December 2003)
17. Eder, J., Kappel, G., Schrefl, M.: Coupling and Cohesion in Object-Oriented Systems. In: Finin, T.W., Yesha, Y., Nicholas, C. (eds.) CIKM 1992. LNCS, vol. 752. Springer, Heidelberg (1993)
18. Feuerlicht, G., Lozina, J.: Understanding Service Reusability. In: The Proceedings of 15th International Conference Systems Integration 2007, Prague, Czech Republic, June 10-12, 2007, pp. 144–150 (2007) ISBN 978-80-245-1196-2
19. Vogel, T., Schmidt, A., Lemm, A., Österle, H.: Service and Document Based Interoperability for European eCustoms Solutions. Journal of Theoretical and Applied Electronic Commerce Research 3(3) (2008) ISSN 0718–1876
20. Papazoglou, M.P., Heuvel, W.V.D.: Service-oriented design and development methodology. International Journal of Web Engineering and Technology 2(4), 412–442 (2006)
21. Baker, S., Dobson, S.: Comparing service-oriented and distributed object architectures. In: Meersman, R., Tari, Z. (eds.) OTM 2005. LNCS, vol. 3760, pp. 631–645. Springer, Heidelberg (2005)
22. Feuerlicht, G.: Service aggregation using relational operations on interface parameters. In: Georgakopoulos, D., Ritter, N., Benatallah, B., Zirpins, C., Feuerlicht, G., Schoenherr, M., Motahari-Nezhad, H.R. (eds.) ICSOC 2006. LNCS, vol. 4652, pp. 95–103. Springer, Heidelberg (2007)
23. Papazoglou, M., Yang, J.: Design Methodology for Web Services and Business Processes. In: Proceedings of the 3rd VLDB-TES Workshop, Hong Kong, pp. 54–64 (August 2002)

24. Ceri, S., Pelagatti, G.: Distributed databases principles and systems. McGraw-Hill Computer Science Series. McGraw-Hill, New York (1984)
25. Fowler, M.: Patterns of Enterprise Application Architecture. The Addison-Wesley Signature Series. Addison-Wesley, Reading (2002); Pearson Education, p. 533, ISBN 13: 9780321127426
26. Kloppmann, M., et al.: Business process choreography in WebSphere: Combining the power of BPEL and J2EE. IBM Systems Journal 43(2), 270 (2004)
27. Bleul, S., Weise, T., Geihs, K.: Making a Fast Semantic Service Composition System Faster. In: The Proceedings of The 9th IEEE International Conference on E-Commerce Technology and the 4th IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services, 2007 CEC/EEE 2007, Tokyo, pp. 517–520 (2007) ISBN 0-7695-2913-5
28. Chen, L., et al.: Towards a Knowledge-Based Approach to Semantic Service Composition. LNCS, pp. 319–334. Springer, Heidelberg (2003)
29. Nierstrasz, O., Meijler, T.D.: Research directions in software composition. ACM Computing Surveys (CSUR) 27(2), 262–264 (1995)
30. Nierstrasz, O.M., et al.: Object-oriented software composition. Prentice Hall, Englewood Cliffs (1995)
31. Kane, K., Browne, J.C.: CoorSet: A Development Environment for Associatively Coordinated Components. LNCS, pp. 216–231. Springer, Heidelberg (2004)
32. Scheben, U.: Hierarchical composition of industrial components. Science of Computer Programming 56(1-2), 117–139 (2005)

A Conceptual Framework for Unified and Comprehensive SOA Management*

Ingo Müller[1], Jun Han[1], Jean-Guy Schneider[1], and Steven Versteeg[2]

[1] Swinburne University of Technology, Hawthorn, Victoria 3122, Australia
{imueller,jhan,jschneider}@swin.edu.au
[2] CA Labs, CA (Pacific), Melbourne, Victoria 3004, Australia
steven.versteeg@ca.com

Abstract. Business requirements, such as regulations and laws, corporate policies, quality of service aspects, etc. affect the management of enterprise-scale SOA systems during different life-cycle stages. Such requirements also induce interdependencies across different elements and aspects of an SOA system. Therefore, a systematic approach to SOA management must be in place in order to effectively govern the development and execution of automated business processes throughout the entire SOA life-cycle in compliance with business requirements. Until now, industry and research have focused on specific management aspects rather than unified and comprehensive solutions. This paper addresses this issue by proposing a conceptual framework for SOA management that combines a micro-kernel/plug-in architecture with the concept of management workflows. The micro-kernel incorporates a unified registry/repository model, facilitating the extension of specific management capabilities with plug-ins. Management workflows compose the capabilities of multiple plug-ins into comprehensive management processes that can be linked to events in different places in an SOA system.

1 Introduction

Service-oriented architecture (SOA) constitutes a set of design principles and best practices for guiding the implementation and execution of automated business processes in heterogeneous IT environments. SOA is a complex design exercise that intrinsically ties together business requirements, software engineering aspects, and operational characteristics of IT infrastructures. It results in a multitude of concrete architectures and implementations varying from small sets of Web services to complex enterprise-scale mediation systems that automate business processes across different application silos and organisational bodies.

The key to enterprise-scale SOA is business/IT alignment. In order to be successful, SOA-based business process automation must yield maximum value for a business at the lowest possible costs with acceptable risks and in compliance with business requirements. Business requirements, including regulations and

* This work is supported by the Australian Research Council and CA Labs.

G. Feuerlicht and W. Lamersdorf (Eds.): ICSOC 2008, LNCS 5472, pp. 28–40, 2009.

laws, corporate policies, quality of service aspects, etc. affect the management of SOA systems during different life-cycle stages. They also induce interdependencies across different elements and aspects of an SOA system. Therefore, a systematic approach to SOA management must be in place to effectively govern the development and execution of automated business processes throughout the entire SOA life-cycle in compliance with business requirements.

According to a recent Gartner press release [1], business/IT alignment is where many SOA projects fail. The study identifies insufficient SOA governance as an "increasingly significant" risk of failure and predicts that by 2010 "less than 25 percent of large companies will have the sufficient technical and organisational skills necessary to deliver enterprise wide SOA." Yet, industry and research have not focused on comprehensive and unified governance solutions. Existing SOA management products are fragmented, separate the management of different assets and life-cycle stages, and focus primarily on engineering aspects of runtime service management. Research contributions are either focused on abstract design guidelines and best practices or on specific management aspects.

Business/IT alignment is crucial for enterprise-scale SOA, but yet little understood. We argue that an essential part of a systematic investigation of SOA governance and management issues is the development of a conceptual software framework that leverages required levels of business/IT alignment by unifying the management of relevant assets across the *entire* SOA life-cycle. This paper proposes such a framework that is aimed at enabling agile, unified, and comprehensive SOA management by combining a micro-kernel/plug-in architecture with the concept of management workflows.

The rest of this paper is organised as follows. Section 2 outlines the fundamental concepts of SOA governance and SOA management as they motivate our framework. Section 3 proposes our conceptual SOA management framework, followed by an outline of the generic principles of management workflows in Section 4. Current trends in industry and research are discussed in Section 5. Finally, Section 6 summarizes this paper and gives an outlook to future work.

2 SOA Governance and Management

The structural openness at the architecture level, the diversity and complexity of enterprise-scale SOA systems, and the mandatory compliance with business requirements require explicit measures for the effective management of SOA systems. These measures are referred to as *SOA governance*. SOA governance can be typified as means to reconcile the business requirements with the characteristics of the existing IT infrastructure that affect an SOA system (Figure 1). This reconciliation or mediation is also denoted as *business/IT alignment*.

The core activities of SOA governance are a *top-down to-be* analysis and a *bottom-up as-is* analysis. The starting point of any SOA initiative must be the clear definition of relevant assets (*e.g.* business services/processes, business policies, etc.) that reflect functional and non-functional business requirements, including aspects such as accountability, ownership, permissions, and compliance to specific regulations, laws, standards, best practices, or quality of service

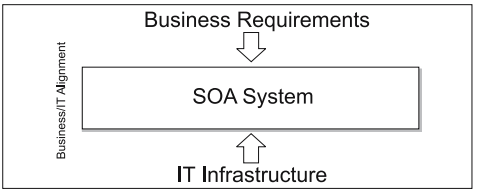


Fig. 1. Illustration of the role of SOA as mediator between business requirements and an existing IT infrastructure

requirements. The identification of business requirements is imperative for SOA because they form the base for the design, implementation, and operation of SOA-based automated business processes. Hence, we assume for the remainder of this paper that all relevant business requirements are specified (in)formally with a business reference model using existing standards such as BPMN or BPEL, and frameworks such as ITIL, COSO, and Cobit.

SOA governance also encompasses an as-is analysis of the static and dynamic properties of the existing IT infrastructure. It is essential to understand and constantly validate the dependencies between business requirements and infrastructure characteristics. Therefore, SOA governance is a continuous process that constantly 'mediates' between business requirements and the operational reality in an SOA system. In that sense, SOA governance establishes a control process on top of an SOA system as shown in Figure 2. The SOA control process puts procedures in place for directing the automation, guiding the execution, and controlling the compliance of business processes with business requirements. In case violations are detected, respective procedures define and trigger mitigation strategies.

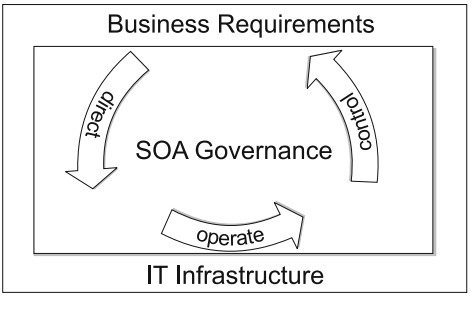


Fig. 2. Illustration of the SOA governance control loop

The implementation and execution of SOA governance procedures with concrete concepts and techniques is denoted as *SOA management*. To ensure effective SOA governance, SOA management must address the following issues:

- *Focus on all relevant assets in an SOA system.* Assets model relevant entities and aspects of an SOA system. For example, business services model

atomic business services and complex business processes. Business actors model existing users and decision makers. Business policies model business requirements. The mentioned assets form only the core set. There are typically more assets in concrete SOA projects.

- *Define and manage the entire life-cycle for every asset.* A simplified SOA life-cycle model contains the four stages: *development*, *deployment*, *execution*, and *evolution*. A life-cycle stage defines a set of artefacts and metadata items that represent an asset. A life-cycle stage also specifies a set of valid operations that can be performed to manipulate the representations of assets. More complex life-cycle models exist. They may vary for different assets.
- *Establish traceability.* The aim of SOA governance is to support business/IT alignment. Therefore, it is essential to manage accountability and compliance of actions that process/manipulate assets in an SOA system (*e.g.* add a service to a repository, enact a business process). Besides the real-time monitoring and validation of such actions and the overall system status, persistent records must be kept in form of logging and audit trails.

Effective SOA management generates large amounts of data (audit trails for tracking actions performed on assets) and includes extensive run-time monitoring (interception and introspection of messages). Therefore, SOA management/governance has a direct impact on the performance of an SOA system, leaving SOA practitioners with a difficult problem. As the Gartner press release [1] points out: "there is no one size fits all" approach to SOA management because "too little or too much governance will kill an SOA project." Accordingly, an SOA management approach must be customisable to specific business requirements and characteristics of an SOA system.

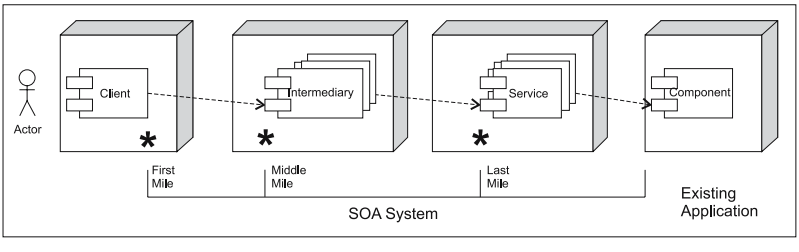


Fig. 3. Simplified UML deployment diagram of a generic enterprise-scale SOA system (The asterisks mark prospective locations for manager components)

Consider a generic enterprise-scale SOA system as depicted in Figure 3. In general, an SOA system can be structured into three sections, denoted as *miles*. The *first mile* spans the section of an SOA system between a client application and a gateway/entry point to the SOA mediation tier. The *middle mile* encompasses a mediation tier of intermediaries (*e.g.* Enterprise Service Bus) that virtualise the location and technology of the actual service components. The *last mile* contains atomic and composite service components that expose and

aggregate the functionality of components of existing applications and software systems. Hence, composite services are an integral part of an SOA system in order to enable the uniform management of atomic services and compositions of services, respectively.

There are a number of requirements for an SOA management approach that can be drawn from the generic layout of an SOA system and the scope of SOA management as outlined before:

1. SOA management is performed in various places in an SOA system. Management facilities are typically incorporated directly at the beginning of every mile of an SOA (as marked with asterisks in Figure 3) to ensure 'end-to-end' management of service requests.
2. SOA management is performed by manager components that consist of three logical parts: (i) a *sensor*, a policy enforcement point (PEP) that intercepts messages, (ii) a *decision maker*, a policy decision point (PDP) that analyses messages according to well-defined policies and rules, and (iii) an *actuator* that triggers actions (*e.g.* message manipulation, event notification, mitigation activities) according to the outcome of the decision making.
3. Manager components perform *local* management of 'what matters and where it matters' in a lightweight fashion to limit the impact on the performance of a managed SOA system to a minimum.
4. Manager components coordinate and synchronise their activities within and across miles that are affected by service requests. A 'single form of record' is required in order to enable manager components to aggregate and share information about assets, their representations, and monitoring/logging data.
5. Manager components can be flexibly adapted to changes of business requirements/the existing IT infrastructure. The separation of business functionality from management functionality and the use of declarative techniques enables non-invasive, extensible, and agile management approaches that reduce the need for extensive coding and recompilation.
6. SOA management approaches are customisable. They can be tailored to the specific scope and requirements of an SOA project. They facilitate the incorporation of various techniques and technologies to solve management aspects in an appropriate manner, *e.g.* utilise management capabilities of the existing IT infrastructure such as federated identity management (to reduce the costs of an SOA project and to capitalise on tested IT infrastructure).

3 Conceptual Management Framework

We propose a generic framework for unified and comprehensive SOA management of enterprise-scale SOA systems based on the requirements outlined above. To meet these requirements, an SOA management system cannot exhibit a monolithic or any other rigid structure that imposes limiting constraints on the target SOA system. Hence, we argue SOA management should be implemented into an SOA system as a sub-system in form of a *hub-spoke architecture* such that the

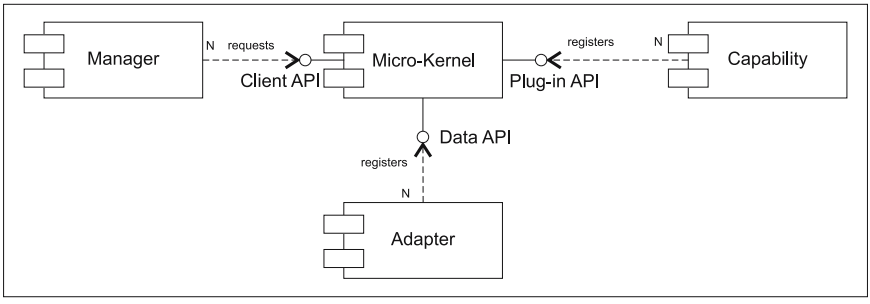


Fig. 4. UML component diagram of the proposed SOA management framework

spokes can be flexibly intertwined with business logic components of the SOA system. Based on the general assertion that service-oriented design principles are good means for developing agile, extensible, and adaptable software systems, we intend to utilise this concept together with a lightweight architectural style in a uniform SOA management framework as shown in Figure 4.

The centrepiece of our SOA management framework is a *micro-kernel* that models a unified *registry/repository* as a 'single form of record'. The micro-kernel establishes a facade to a shared data space of an enterprise-scale registry/repository. It provides uniform access to information about all relevant assets and their corresponding representations from disparate sources in an SOA system and, for example, enables a comprehensive impact analysis of changes to any relevant asset. Although the micro-kernel represents the central hub component in the hub-spoke architecture, it can be implemented in a distributed manner. The micro-kernel implements functionality (Data API) to (i) register/de-register adapters to access databases/data sources, and their respective data models, (ii) handle discrepancies in the various data models with the help of adapters, and (iii) enable uniform data query and manipulation operations.

The organisation of the registry/repository is oriented on the definition of the ebXML Registry/Repository standard [2] which is aimed at managing any kind of electronic content, that is, XML documents, text files, and arbitrary binary content. An instance of electronic content that represents the whole or parts of an asset (also denoted as *artefact*) is stored in the *repository* section. Moreover, *metadata* (associated with artefacts) is stored in the *registry* section. Accordingly, the repository is a sink for artefacts whereas the registry embodies a cataloguing system that keeps standardised metadata for enabling the effective management and manipulation of the repository content.

The micro-kernel does not provide SOA management functionality other than for querying and manipulating artefacts and metadata items, respectively. Actual management functionality must be implemented as *management capabilities* and must be connected to the micro-kernel as *plug-ins*. Hence, the micro-kernel provides functionality (Plug-in API) to (i) register/de-register plug-ins, (ii) discover and match management capabilities, and (iii) invoke management capabilities.

Management capabilities can access and manipulate artefacts and metadata items in the registry/repository through the Plug-in API of the micro-kernel. The permissions to do so depend on the properties of the actor whose service request triggered a management action, and the affected assets and their state. Permissions are verified by a specific management capability prior to execution. Management capabilities may be of different granularity and complexity ranging from simple event notification, template validation, or the creation of an audit trail record to the integration of external IT management systems (*e.g.* for utilising federated identity management).

The micro-kernel also provides functionality (Client API) for *manager components* to request management capabilities for supporting the monitoring, enforcement, and tracking of the compliance of service provisioning activities with business requirements. Manager components are situated in the SOA system infrastructure next to the service components they manage. They intercept messages to and from the managed service components in a transparent fashion. A manager component comprises of three logical units that provide functionality for different aspects of SOA management: (i) intercepting messages, (ii) performing decision making, and (iii) executing actions on messages and/or in the SOA system. The logical units of a manager component may be distributed depending on particular management aspects, specific business requirements, and the characteristics of the managed SOA system.

Manager components are typically situated in every mile of an SOA system. First mile manager components provide capabilities to client applications *e.g.* to dynamically discover services, negotiate and fix SLAs, validate, manipulate, and transform messages to meet security requirements. They may be embedded into a remote API in order to be installed on client computers outside the actual SOA infrastructure. They are crucial for managing SLAs because they are located closest to the service consumers and, therefore, are most suitable to assess the consumer experience. There is typically a *1-to-1* relationship between client applications and first mile manager components.

Middle mile manager components are located in the mediation tier of an enterprise-scale SOA system which means either at a gateway to the mediation tier or in a container with intermediary components. They are primarily concerned with authentication, authorisation, security, and performance aspects and may be involved in message transformation and routing activities. They may also monitor type and number of specific service requests in order to provide statistical information for optimised routing and utilisation of infrastructure elements. There is typically a *1-to-1* or *1-to-n* relationship between middle mile manager components and intermediary components.

Last mile manager components are placed in container with the service component they monitor. They enforce the enactment of business processes and manage consequential effects (*e.g.* logging statistical data, creating audit trail records, sending event notifications to subscribers) in compliance with relevant business requirements, standards, and technologies that are implied by disparate development teams, existing IT infrastructures, organisational bodies, etc. They

are also responsible for triggering appropriate mitigation actions in case existing software systems and applications that are represented by services malfunction or violate particular business requirements. There is typically a *1-to-1* or *1-to-n* relationship between last mile manager components and service components.

Manager components operate based on *management workflows*. Management workflows represent procedural context-aware knowledge about how to apply and enforce business requirements for the enactment of business services/processes with respect to the properties of the requesting actor. Management workflows can be understood as the glue that ties together relevant business requirements and activities in an SOA system for the implementation and execution of automated business processes. As such, management workflows are the technical means that implement SOA governance procedures in order to ensure the 'direct-operate-control' SOA governance control loop. Management workflows utilise the components of the SOA management framework by specifying compositions of management capabilities that define how business requirements are enforced, for example, what information of intercepted messages is required, how it is processed, and what actions are triggered depending on the outcome of decision making processes. The use of declarative techniques for defining management workflows increases the agility of our SOA management approach because it allows the flexible adaptation to changes without coding efforts, re-compilation, or re-starting of affected manager components.

4 Outline of Management Activities

This section outlines the generic principles of how the management workflow concept integrates in an SOA system. For the sake of simplicity we illustrate the intertwined processing of service requests and management workflows with a simplified scenario in which one manager component is assigned to exactly one service component as shown in Figure 5.

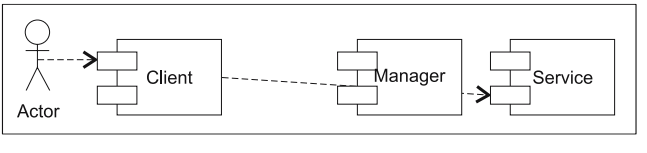


Fig. 5. Example scenario with one client, one manager, and one service component

The management of a request of an atomic service works as follows:

1. An actor triggers a service request message that is sent by the client to an appropriate service component.
2. Before the message arrives at the service component it is intercepted by its associated manager component in a transparent fashion.
3. The manager component inspects header/payload of the message in order to determine its context (properties of actor/requested services). Based on the

Table 1. Sequential management workflow for an incoming service request

Task	Description
1	Retrieve service requester identity and obtain credentials/permissions.
2	Verify requester credentials against security and access policies.
3	Verify request message format based on an XSD template.
4	Validate request message contents according to pre-conditions, e.g. semantics of parameter values and value ranges.

retrieved information, the manager component selects a suitable management workflow from the registry/repository that encompasses management tasks for all relevant business requirements (cf. Table 1 for a simple example).
4. The manager component enacts the management workflow using the message as input. The tasks of the management workflow process data from the message and may alter, add, or remove data from the message. Upon execution, a management task raises a flag in case a policy is violated.
5. If all management tasks have been successfully executed, the message is sent to the service component. If a policy is violated, the manager component does not send the message to the service component but generates and returns an error message to the client.
6. The service component executes the requested business service and produces a response message that is subsequently sent back to the client.
7. The response is also transparently intercepted by the manager component which then performs steps 2 - 4 on the response message according to another management workflow (cf. Table 2 for a simple example). If all tasks of the management workflow have been successfully executed, the manager component passes on the response message to the client. In case of failure, an error message is generated and returned to the client.

Table 2. Sequential management workflow for an outgoing service response

Task	Description
1	Create and store audit trail record.
2	Annotate response message with performance metrics information.
3	Send event notifications to subscribed listeners.

The manipulation of messages with annotations is a powerful means for the manager component to provide additional information or to correct the header/payload of a message to guide the processing of service requests. Message annotations are also a powerful means for exchanging information or coordinating management activities among distributed manager components across and within first/middle/last miles in more complex scenarios.

Hence, message annotations are essential for coordinating the activities between different manager components involved in the management of business processes. Firstly, because we do not assume the existence of additional communication channels between management components. Secondly, because the

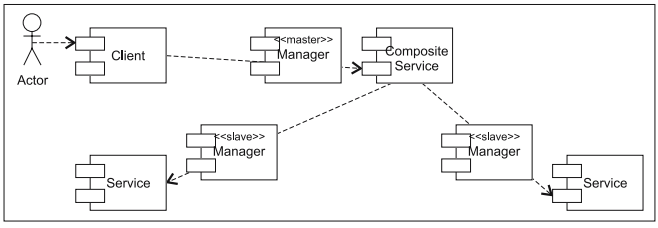


Fig. 6. Relationship between manager components and services in a composite service

relationships between manager components are dynamic depending on specific service requests, message routing, etc. Therefore, the *master* manager component associated with a composite service component coordinates the activities of *slave* manager components associated with component services via message annotations (cf. Figure 6).

5 Trends in Industry and Research

The topic of SOA management and governance marks an active area of development in industry and research in academia. The vendors of SOA infrastructure products have transformed their offerings in the past 5 years from disparate development tools and simple UDDI registries into comprehensive SOA infrastructure and management suites. The key driver of this evolution is the need for products to support close loop management and closer business/IT alignment in order to be beneficial for practitioners. Despite the progress, industry products still expose a number of open issues.

Firstly, full interoperability among and within product suites for truly unified SOA management is not available due to disparate tool sets and standards (vendor lock-in) and growth by acquisition strategies (*e.g.* IBM SOA foundation includes products of the WebSphere, Rational, and Tivoli product families). Various product suites comprise heterogeneous tools of partially overlapping functionality and/or limited interoperability. Thus, manual workarounds are often required that prevent the fully systematic and automated management of assets throughout all life-cycle stages.

Secondly, although a 'single form of record' service registry/repository is typically described as the pivotal element for effective SOA management (*e.g.* Strnadl [3]) existing products suites incorporate multiple disconnected local repositories. Moreover, most product suites separate service registry (based on the limiting UDDI standard) from metadata repositories resulting in the poor utilisation of metadata for service discovery and impact analysis.

Thirdly, existing product suites support the aggregation of management capabilities in the form of simple approval/promotion workflows that cannot be substantially extended. As a consequence, different management aspects are typically handled separately.

Fourthly, existing product suites are primarily focused on engineering aspects of service run-time management and policy enforcement. They leverage only limited business/IT alignment in form of business activity monitoring (BAM) that allows the definition and monitoring of business-level key performance indicators (KPI). In general, the systematic mapping between business-level aspects and engineering-level models and methods is still not well understood. The expressiveness of BAM is currently restricted to simple metrics and decision making/decision support. Policy management is concentrated on engineering aspects for implementing security, reliability, and transaction management.

Research has also shown a great deal of interest in SOA management and governance. Papazoglou *et al.* [4] lists the topic of service management and monitoring as one of four key research topics in the area of SOC. And, there already exists a significant body of research results. Firstly, research work investigated specific management capabilities. The interested reader is referred to Papazoglou *et al.* [4] for an overview.

Secondly, research work was conducted on general conceptual frameworks that are aimed at guiding practitioners in designing, implementing, deploying, and operating SOA systems. Although, SOA management is an intrinsic part of these conceptual frameworks, management aspects are not explicitly investigated as discussed by Arsanjani *et al.* [5]. Moreover, these conceptual frameworks typically remain at a high level of abstraction. As an example, consider the business/enterprise architecture level by Schepers *et al.* [6] and Roach *et al.* [7].

Thirdly, a few research activities address the problem of SOA management with the same scope as we do. Arrott *et al.* [8] describes an extended service component concept denoted as *rich service*. A rich service models functional and non-functional aspects of one business service/process. It can be flexibly managed and extended via its internal message bus and plug-in mechanism. However, the focus of this concept on the component level restricts the modelling and coordination of management activities across components. In contrast, our approach is focused on the system level. Siljee *et al.* [9] outlines DySOA which is a management infrastructure that links a set of management components (similar to our manager components) to every service component in an SOA system. DySOA is focused on QoS-driven run-time management of services. Unlike our approach, it does not promote the unified management of all relevant assets throughout the full SOA life-cycle. Belter's work [10] provides a generic high-level framework for service management systems that is centred on a UDDI service registry. The framework is focused on comprehensive management of all relevant SOA assets. However, our framework is more specific. Belter's contribution omits information about how elements of the management system interleave with components of the managed SOA system, how management capabilities are implemented, extended, adapted, and how business policies are enforced.

The latest EU Research Framework Programme (`www.cordis.europa.eu/fp7`) addresses the current lack of research of the complex problems of business/IT alignment and SOA management/governance by providing significant

funding for a number of large projects including Compass (www.compas-ict.eu), Master (www.master-fp7.eu), and S-Cube (www.s-cube-network.eu).

6 Summary and Future Work

Industry and research have until now only focused on specific management aspects rather than unified SOA management. We argue that a unified and systematic approach to SOA management is essential to effectively govern enterprise-scale SOA systems in compliance with business requirements. Therefore, we have proposed a conceptual framework for fostering unified and comprehensive SOA management as part of a systematic investigation of SOA governance and management issues. Our framework is characterized by its flexible micro-kernel/plug-in architecture and the concept of management workflows which enables agile context-based management of assets such as business services/processes, policies, and actors. In future work, we plan to verify our framework by investigating specific management issues and incorporating the results into the framework based on our expertise in configurable adaptive systems [11], software composition [12], and policy-based management of service registries [13] in order to create techniques and tools that support practitioners in their daily work.

References

1. Goasduff, L., Forsling, C.: Bad Technical Implementations and Lack of Governance Increase Risks of Failure in SOA Projects. Online Press Release (June 2007), http://gartner.com/it/page.jsp?id=508397
2. Breininger, K., Farrukh Najmi, N.S.: ebXML Registry Services and Protocols, Version 3.0 (2005), http://docs.oasis-open.org/regrep/regrep-rs/v3.0/regrep-rs-3.0-os.pdf
3. Strnadl, C.F.: Bridging Architectural Boundaries Design and Implementation of a Semantic BPM and SOA Governance Tool. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749, pp. 518–529. Springer, Heidelberg (2007)
4. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F.: Service-Oriented Computing: a Research Roadmap. International Journal of Cooperative Information Systems 17, 223–255 (2008)
5. Arsanjani, A., Zhang, L.J., Ellis, M., Allam, A., Channabasavaiah, K.: S3: A Service-Oriented Reference Architecture. IEEE IT Professional 9, 10–17 (2007)
6. Schepers, T., Iacob, M., Van Eck, P.: A Lifecycle Approach to SOA Governance. In: Proceedings of the 2008 ACM Symposium on Applied Computing (SAC 2008), Fortaleza, Brazil, pp. 1055–1061. ACM Press, New York (2008)
7. Roach, T., Low, G., D'Ambra, J.: CAPSICUM - A Conceptual Model for Service Oriented Architecture. In: Proceedings of the 2008 IEEE Congress on Services - Part 1 (Services), Honolulu, USA, pp. 415–422. IEEE Computer Society Press, Los Alamitos (2008)
8. Arrott, M., Demchak, B., Errnagan, V., Farcas, C., Farcas, E., Krüger, I.H., Menarini, M.: Rich Services: The Integration Piece of the SOA Puzzle. In: Proceedings of the IEEE International Conference on Web Services (ICWS 2007), Salt Lake City, USA, pp. 176–183. IEEE Computer Society Press, Los Alamitos (2007)

9. Siljee, J., Bosloper, I., Nijhuis, J., Hammer, D.: DySOA: Making Service Systems Self-adaptive. In: Benatallah, B., Casati, F., Traverso, P. (eds.) ICSOC 2005. LNCS, vol. 3826, pp. 255–268. Springer, Heidelberg (2005)
10. Belter, R.: Towards a Service Management System in Virtualized Infrastructures. In: Proceedings of the IEEE International Conference on Services Computing (SCC 2008), Honolulu, USA, pp. 47–51. IEEE Computer Society Press, Los Alamitos (2008)
11. Coleman, A., Han, J.: Coordination systems in role-based adaptive software. In: Jacquet, J.-M., Picco, G.P. (eds.) COORDINATION 2005. LNCS, vol. 3454, pp. 63–79. Springer, Heidelberg (2005)
12. Lumpe, M., Schneider, J.G.: A Form-based Metamodel for Software Composition. Journal of Science of Computer Programming 56, 59–78 (2005)
13. Phan, T., Han, J., Schneider, J.G., Ebringer, T., Rogers, T.: Policy-Based Service Registration and Discovery. In: Meersman, R., Tari, Z. (eds.) OTM 2007, Part I. LNCS, vol. 4803, pp. 417–426. Springer, Heidelberg (2007)

A Metrics Suite for Evaluating Flexibility and Complexity in Service Oriented Architectures

Mamoun Hirzalla[1], Jane Cleland-Huang[2], and Ali Arsanjani[3]

[1] DePaul University and IBM
mhirzall@cs.depaul.edu
[2] DePaul University
jhuang@cs.depaul.edu
[3] IBM
arsanjan@us.ibm.com

Abstract. Service Oriented Architecture (SOA) is emerging to be the predominant architectural style of choice for many organizations due to the promised agility, flexibility and resilience benefits. However, there are currently few SOA metrics designed to evaluate complexity, effort estimates and health status of SOA solutions. This paper therefore proposes a SOA metrics framework which includes both service level and SOA-wide metrics to measure design and runtime qualities of a SOA solution. The SOA-wide metrics predict the overall complexity, agility and health status of SOA solutions, while service level metrics focus on the fundamental building blocks of SOA, i.e. services. The combined views deliver a compelling suite of SOA metrics that would benefit organizations as they consider adopting SOA. These metrics, which are based on observations of many SOA engagements, are illustrated through a case study that describes a recent ongoing project at IBM where SOA was utilized to build the solution assets.

Keywords: SOA metrics, SOA complexity, agility, flexibility, SOA health.

1 Introduction

Service Oriented Architecture (SOA) is becoming an increasingly popular architectural style that focuses on providing the right tools and methods for building distributed applications. The fundamental building blocks of SOA are repeatable business tasks realized as services and implemented in a variety of distributed components such as CORBA, EJBs and web services [9].

From a business perspective, one of the primary objectives of SOA-based systems is the alignment between business and IT and the flexibility and business agility that SOA injects into an organization [9]. This is achieved through systematically designing and building a SOA-based solution using a method such as SOMA, invented and initially developed in IBM [7]. SOMA defines key techniques and provides prescriptive tasks and detailed normative guidance for analysis, design, implementation, testing, and deployment of services, components, flows, information, and policies needed

G. Feuerlicht and W. Lamersdorf (Eds.): ICSOC 2008, LNCS 5472, pp. 41–52, 2009.

to successfully design and build a robust and reusable SOA solution in an enterprise [7]. However SOMA, like other SOA-based methodologies, does not provide techniques or metrics for measuring underlying complexity and flexibility qualities of a SOA-solution.

This paper therefore proposes a new suite of metrics, designed specifically to evaluate flexibility and agility versus complexity of a SOA solution. The set of metrics can be used to provide a diagnosis for the health of a SOA solution by providing information about services used within a SOA solution, their composition interfaces or provided operations, architectural decisions, flexibility, agility and complexity. The term "health" of SOA solutions is not limited to the metrics or characteristics identified in this paper. Obviously, there are other elements that must be considered to determine the overall health of a SOA solution. SOA Governance with its emphasis on security, management and testing is a major factor to consider when evaluating such a question. This paper will focus on the architectural considerations as they pertain to the "health" factor.

The proposed SOA metrics are grouped into two major categories: design-time SOA metrics and run-time SOA metrics. Table 1 summarizes the metrics categories and their applicability.

Table 1. Individual Design-time and Run-time SOA Metrics

Classification	Metric Name	Metric Applicability
Design-time	Weighted Service Interface Count (WSIC)	Service
Design-time	Stateless Services (SS)	Service
Design-time	Service Support for Transactions (SST)	Service
Design-time	Number of Human Tasks (NHT)	Service
Run-time	Number of Services (NOS)	SOA Solution
Run-time	Service Composition Pattern (SCP)	SOA Solution
Run-time	Service Access Method (SAM)	SOA Solution
Run-time	Dynamic vs. Static Service Selection (DSSS)	SOA Solution
Run-time	Service Realization Pattern (SRP)	Service
Run-time	Number of Versions Per Service (NOVS)	Service

These metrics were identified as a result of experiences gained from engaging in building numerous SOA solutions. More traditional object-oriented (OO) metrics can also be used to evaluate the internal complexities of individual services [6]. Although, each metric is individually reported, results are also aggregated in terms of three indices, a general SOA Complexity Index (SCI), a Services Complexity Index (SVCI), and a Flexibility and Agility Index (FAI). Table 2 provides a list of the proposed SOA indices and their descriptions.

These metrics can be captured as part of the SOMA lifecycle. Design-time metrics are gathered during the identification, specification, and realization phases, while runtime metrics are gathered during implementation and deployment. Furthermore, metrics collection is not difficult, because metrics can be automatically collected through inspecting SOA artifacts such as a service's Web Service Description Language (WSDL) file, Choreography Description File (CDL), related policies documents, or through inspecting SOAP messages and source code.

The remainder of this paper is laid out as follows. Section 2 provides a survey of SOA metrics, and explains why OO metrics are useful yet insufficient for measuring

SOA solutions. Section 3 introduces and describes SOA design-time metrics. Section 4 introduces and describes SOA runtime metrics. Section 5 demonstrates the usefulness of the metrics through a case study taken from a recent ongoing project that utilized SOA to build the solution assets. While this case study does not empirically validate the proposed metrics, it illustrates how they could be reasonably used to evaluate the complexity and agility of a SOA solution and provide some indications regarding the health of a SOA solution. Finally, section 6 concludes with a discussion for future work.

Table 2. Aggregate SOA Indices

SOA Index	Description
SOA Complexity Index (SCI)	Measures the inherent complexity of the SOA solution including its security, management and SOA governance measures, all of which offer significant benefits but also increase the complexity of the overall SOA
Services Complexity Index (SVCI)	Measures complexity, but looks at the individual complexities of each of the composed services
Flexibility and Agility Index (FAI)	Tracks the flexibility and agility of the SOA solution, which represent SOA's primary objective to bring business agility and flexibility to an organization [4]

2 Background Information

Measurement is an important component of any process improvement initiative. Software metrics enable qualities of interest to be measured and evaluated, in order to identify potential problems, and to provide insight into the costs and benefits of a potential solution. Unfortunately current SOA metrics are relatively immature and tend to suffer from many of the problems previously identified by Chidamber and Kemerer in respect to early OO metrics. These problems include lack of desirable measurable properties, over generalization, focus on specific technologies, and collection difficulty [6].

Although OO metrics [6] can be used to measure the internal complexity of a service built on the OO paradigm; they are not sufficient for measuring more global SOA qualities. A few researchers have proposed various SOA metrics. For example, Liu et al. [2] developed complexity and attackability metrics and showed that complexity has a negative impact on security. Their Average Service Depth metric computes the average number of dependency relationships per atomic service node, as representatives of various software capabilities within a system. Rud et al. [1] focused on the infrastructure and performance aspects of SOA solutions and identified many SOA metrics that are granular in nature. These metrics were classified into the three major areas of complexity, criticality and reliability, and performance metrics. They identified a relationship between complexity of a service and amount of time required to build such a service. Qian et al. [3] developed decoupling metrics for SOA software composition such as Average Service State Decomposition (ASSD), Average Service Persistent Dependency (ASPD) and Average Required Service Dependency (ARSD), and used it to evaluate decoupling between service-oriented components in

the service composition such as Business Process Execution Language (BPEL); a useful set of metrics that should be considered for loose coupling considerations as part of the health status of SOA solutions.

Unfortunately, none of these metrics provide a comprehensive approach for measuring flexibility and agility which represent significant factors in the short and long-term success of a SOA solution. In contrast, the metrics proposed in this paper are specifically designed to evaluate the impact of SOA architectural decisions upon the flexibility, agility and complexity of a SOA solution.

3 SOA Design-Time Metrics

The four metrics defined in this section measure the flexibility, agility, and complexity of the solution in respect to design time decisions. The metrics are independent of the underlying code, and could be applied to either JEE web services or .Net services. Metrics are first computed for individual services and then compiled into a more global metric and applied to the FAI, SVCI, and SCI indices.

Metric 1: Weighted Service Interface Count (WSIC)

Definition: WSIC = The weighted number of exposed interfaces or operations per service as defined in the WSDL documents. The default weight is set to 1. Alternate weighting methods, which need to be validated empirically, can take into consideration the number and the complexity of data types of parameters in each interface. In the default case, WSIC simply returns a count of the number of exposed interfaces or methods defined in the WSDL documents.

Hypothesis: The higher the number of service interfaces the more complex a service becomes and by association the more complex a SOA solution becomes. In addition, there is a direct relationship between the complexity of the exposed interfaces and the complexity of the data structures required per interface.

Observations: The greater the number of defined interfaces per service within a SOA solution the more complex a service becomes due to the following factors. (i) The amount of work required to specify, construct and test every interface on the service increases. (ii) The amount of monitoring required to ensure that service level agreements (SLAs) are met increases with every invocation of an interface. (iii) With the increase in complexity of individual interfaces of the data structures for a given service, performance and problem determination concerns may become a primary issue. Performance and root cause issues are hard to predict and diagnose.

Impact on defined indices: Both SVCI and SCI increase as WSIC increases. There is no impact on the FAI index.

Metric 2: Stateless Services (SS)

Definition: SS = The fraction of services which are stateless (SLS) as opposed to stateful (SFS) as defined in the Web Services Resource Framework (WS-RF) [8] or WS-Context [12]. SS = SLS / (SLS + SFS).

Hypothesis: Developing stateful web services is much harder than developing stateless web services and therefore increases the complexity of a given service.

Observations: Both WS-RF and WS-Context define how to support stateful interactions while using the web services programming model. WS-RF follows a resource-based approach to support state, while WS-Context uses a context that resembles shared state management across different interacting web services. Regardless of which approach is used, supporting transactions in web services will add an additional layer of complexity to programming web services. Complexity also increases with an increase in the number of web services that are participating in a stateful interaction.

Impact on defined indices: SVCI increases as the raw count of SFS services increase. Both SCI and FAI increase with decreased SLS values, i.e. when the fraction of stateless services increases. SCI would increase by a higher value if SLS fraction of stateless services decrease compared to stateful services.

Metric 3: Service Support for Transactions (SST)

Definition: SST = The fraction of transaction-aware services (TAS) in relation to the overall number of transaction-aware and non-transaction aware (NTAS) services within the SOA solution. SST = NTAS/(NTAS+TAS)

Hypothesis: Web services supporting transactions are more complex to build and as a result increase the overall complexity of SOA solutions.

Observations: Traditional transaction systems use a two-phase commit protocol to achieve atomicity between transaction participants. Support of transactions in web services is accomplished through support of the WS-TX specification which includes the WS-Coordination, WS-Atomic Transaction and WS-Business Activity specifications [10]. The WS-TX specification requires additional code that needs to be included in the body of a web service and its invoking client. In order to maintain consistency of transactions, compensating transactions are required to provide correct compensation actions. Furthermore, the coordination between transaction-aware services requires additional effort that injects additional complexity into building transaction-aware services and SOA solutions.

Impact on defined indices: SVCI increases as the raw count of transaction-aware services increase. SCI increases as SST values increase, i.e. when the fraction of transaction-aware services increases. FAI is relatively unaffected by support for transactions in web services. However, extensive use of transactions is likely to constrict how flexible and agile a SOA solution becomes.

Metric 4: Service Realization Pattern (SRP)

Definition: SRP = The fraction of services that are realized through Indirect Exposure (IE) in respect to the total number of services that are realized using both IE and Direct Exposure (DE). SRM = IE /(IE+DE)

Hypothesis: The more indirect exposure service realizations in SOA solution, the more complex a service becomes and by association the more complex a SOA solution becomes.

Observations: There are many service realization patterns that can be used for exposing and using services including the two primary patterns of Direct Exposure (DE) and Indirect Exposure (IE). DE refers to exposing current IT systems or modules as a service without having to go through an intermediary component. For example, a stored SQL procedure could be turned into an information service directly by wrapping it through a web service and exposing the web service to consuming clients. Indirect Exposure, on the other hand, refers to exposing current IT systems or a module as a service by going through an intermediary component such as an EJB. Direct Exposure services provide a much faster method for creating and invoking services. They also require less time to decide on appropriate interfaces since they tend to match the interfaces that can be exposed from the legacy asset. Direct Exposure services also require less time to develop and test due to the direct connectivity with the backend system. In comparison, Indirect Exposure realization of services entails additional IT components to mediate between a service and an IT asset. While this provides additional flexibility to the overall SOA solution, it also increases the time to build and test such services, and requires additional management and monitoring steps to ensure services and their associated components are functioning properly.

Impact on defined indices: SVCI increases with the use of IE realization of services and decreases with the use of DE realization. SCI will increase with the increase in the value of the SRP. The higher the ratio of DE to IE realizations, the less complexity. This is inversely related to the ratio of IE services to the overall number of both DE and IE services. In other words, the lower the ratio, the less complexity. FAI depends on the ratio of DE services to the overall number of both DE and IE services. The lower the ratio, the more flexible and agile a SOA solution will be. This is inversely related to the ratio of IE services to the overall number of both DE and IE services. In other words, the lower the ratio, the less complexity.

Metric 5: Number of Human Tasks (NHT)

Definition: NHT = The fraction of tasks as part of a business flow that are manual.

Human Tasks (HT) in a process flow are important due to their need in real life scenarios. For example, a business process flow can invoke many services to automate a banking process that requires the final verification of an auditor through a human interaction. The judicious use of human tasks within a process is accepted as a fact of life. However, with the increased use of human tasks, we end up with less flexible processes within a SOA solution. NHT is computed as NHT = HT/(HT+AT) where AT refers to an automated task.

Hypothesis: The use of too many Human Tasks in a SOA solution increases complexity and decreases flexibility of a SOA solution.

Observations: BPEL defines business processes as collections of activities that invoke services. BPEL doesn't distinguish between services that are provided by applications and other interactions, such as human interactions. From our metrics perspective, any human task that is being executed as part of a BPEL flow within a SOA solution will have an impact on both complexity and flexibility.

Impact on defined indices: There is no direct impact on the service complexity due to the use of human tasks. However, the maintenance issues associated with manual tasks will increase the overall complexity of such tasks and therefore increasing the overall value of the SCI. In addition, too many human tasks will negatively impact the flexibility and agility of the overall SOA solution and decreases the value of the FAI.

4 SOA Run-Time Metrics

SOA run-time metrics measure the overall complexity and agility of a SOA solution. As a result, SOA metrics help in achieving the overall objective of exposing overly complex SOA architectures and provide insights into the flexibility factor of a SOA solution. In order to calculate the SOA run-time metrics, multiple SOA components are considered such as ESB, registry, and SOA governance.

Metric 6: Number of Services (NOS)

Definition: NoS =Total number of services that comprise a SOA solution

Observations: Greater numbers of services increase the complexity of a SOA solution due to the following factors. (i) An increase in the amount of work required to identify, specify, create, test, deploy and maintain such services. (ii) The need to provide better SOA Governance and service lifecycle management for the services within a SOA solution becomes more critical as the number of services used within a SOA solution increase. (iii) The increased number of services within a SOA solution usually places additional demand on the SOA infrastructure where services are deployed to meet service level agreements (SLAs) and scalability requirements.

Hypothesis: The higher number of services within a SOA solution, the more complex a SOA solution becomes.

Impact on defined indices: SVCI is not impacted. SCI increases as the number of services increase.

Metric 7: Service Composition Pattern (SCP)

Definition: SCP = The fraction of web services which are composite.
An Atomic Service (AS) is a web service that can stand on its own and does not require the use of other web services to complete its functionality. In contrast Composite Services (CS) are composed of other web services through aggregation, also referred to as structural composition [12] (CSs), or at runtime through invocation and orchestration of services through use of a workflow (CSwf). SCP is computed as SCP = CS / (AS + CSs + CSwf) and can be further refined according to the composition method. SCPs measures the fraction of composite services constructed using aggregation, while SCPwf measures the fraction composed at runtime. These two metrics are defined as follows: SCPs = CSs / (AS + CSs) and SCPwf = CSwf / (AS + Cswf).

Hypothesis: The use of composite services in a SOA solution increases both complexity and flexibility of a SOA solution.

Observations: Service orchestration refers to the collaboration among services that is driven by a central component or workflow engine, while service choreography refers

to the collaboration among different parties that are each responsible for one or more steps of service interaction. The complexity of choreography stems from the fact that no central entity controls the process as a whole and therefore gleaning insight into the overall picture of the process status requires additional effort. Managing the state of service choreography often leads to complications due to the many events that need to be triggered and correlated to ensure proper execution of business functionality.

Impact on defined indices: There is no impact on the SVCI for an atomic service. However, structural composition will increase the value of the SVCI. An increase in SCP leads to an increase in both FAI and SCI.

Metric 8: Service Access Method (SAM)

Definition: SAM = The fraction of services accessed using a virtualization layer, referred to as Virtualized Access Services (VAS), in respect to the total number of services that are VAS or accessed point to point(PPS). SAM = VAS / (VAS + PPS)

Hypothesis: Virtualized access to services within a SOA solution increases the flexibility and agility of a SOA solution but also increases its complexity.

Observations: Services can be accessed directly by an invoking client or through a broker component, referred to as an Enterprise Service Bus (ESB) which looks up the address of required services through a registry component, retrieves the Web Service Definition Language (WSDL) file, and then binds to that service during the invocation process. The ESB in essence provides a virtualization layer so that invoking clients do not need to know individual physical addresses of services. The ESB is responsible for routing and translating requests and responses among service requestors and service providers. The method of invocation will be referred to as Virtualized Access Services (VAS). The invocation of services also plays a role in the level of complexity associated with this metric. Services that are invoked directly are considered point to point (PPS) connections and are harder to maintain. On the other hand, services invoked through an ESB are easier to maintain but more complex to setup, because adding an ESB component to the overall SOA solution is not a simple task. It requires proper planning and design of the ESB and interacting services. The inclusion of a service registry is not considered a factor for this metric since it is dependent on the level of SOA governance and management required as part of the overall SOA solution, and is likely to be introduced when the number of services exceeds a threshold level.

Impact on defined indices:

There is no impact on SVCI. Over the long-term SCI decreases as SAM increases, i.e. use of point to point connections increase. FAI also decreases as SAM increases.

Metric 9: Dynamic vs. Static Service Selection (DSSS)

Definition: DSSS = The number of services that are selected dynamically (DS) over the total number of services that are selected dynamically or statically (SS).
DSSE = DS/(DS+SS).

Hypothesis: The more dynamic selection of services for execution within a SOA solution the more complex, flexible and agile a SOA solution becomes.

Observations: Service consumers can invoke services through the typical stub and tie classes that are generated by the available tools in the market place. However, there are instances where the business logic requires a truly dynamic method for invoking the proper service based on a business rule that mandates a better service for a platinum-level customer to maintain their loyalty. In such scenarios, having a broker in the middle and the proper advertising of available services becomes mandatory. For this kind of scenario, the ESB and registry along with service monitoring software will play a significant role. However, it increases the overall complexity of maintaining a SOA solution. As Dynamic service invocation provides better flexibility and agility since a process can adapt much quicker than a process those hard codes static locations of services.

Impact on defined indices: There is no impact on SVCI. The higher the ratio of SSE, the higher the complexity of a SOA solution; SCI therefore increases as SSE increases. Moreover, the higher the ratio of SSE, the more flexible a SOA solution.

Metric 10: Number of Versions per Service (NOVS)

Definition: NOVS = The total number of versions over the total number of services within the SOA solution. NOVS = VERSIONS / SERVICES

Hypothesis: The higher the number of service versions the more complex a service becomes and by association the more complex a SOA solution becomes.

Observations: The number of versions available in production per service is dependent on the level of change that services undergo while they are in production. It may signal an unstable service interface in the first place and a situation where services were rushed into production. The greater the number of available versions per service within a SOA solution the more complex a service becomes due to the following factors. (i)The amount of work required to keep track of service versions and their associated clients. Multiple service versions may also provide different SLAs for different consumers. The SOA Governance and management aspects of tracking service versions will become harder with every new service version that is maintained. (ii) The amount of regression testing required per service version increases if a common defect is discovered in one of the service versions. Therefore, additional time is required in order to ensure that all service versions are operating uniformly for similar business logic. The proliferation of many versions for the same service may point to lack of proper design that considers the level of reuse required. Reusable services tend to be more carefully planned, designed and implemented.

Impact on defined indices: Both SVCI and SCI increase as NVS increases. There is no impact on FAI.

5 Case Study

The metrics described in this paper still need to be empirically evaluated across multiple SOA applications to determine if they are complete and relatively non-redundant. In this section we briefly describe their use in a case study which is representative of multiple internal projects developed at IBM. The customer in the case study is a bank that is considering a SOA solution to address some of the primary pain points related to its Account Open business process. The current process was riddled with manual tasks

that were causing the Account Open process to extend to 14 days instead of 10 minutes as offered by the bank's primary competitors.

One of the primary business objectives of the bank's SOA Solution was to inject agility and flexibility into the Account Open business process while trying to minimize complexity of the overall SOA solution. At a high level, the Account Open business process consisted of two primary sub-processes: Account Verification and Account Activation.

SOMA was used to model the optimized business process and identify and specify the services needed to realize the new business process vision. Candidate services were identified using SOMA's top down and bottom up approaches and SOMA's service litmus test (SLT) was applied to rationalize and verify candidate services. These services were developed from existing assets or else programmed from scratch. The SOA solution was built in three iterations. The first iteration of *service creation* focused on creating new services that were used to automate previous manual tasks such as credit check and address validation. The second *service connectivity* iteration focused on integrating disparate systems within the bank through the incorporation of an ESB to virtualize access to services and enhance the overall flexibility of the solution. Finally the *Interaction and Collaboration* iteration provided an enhanced user interface to the web channel by incorporating the automated steps of the newly optimized Account Open business process with the created services.

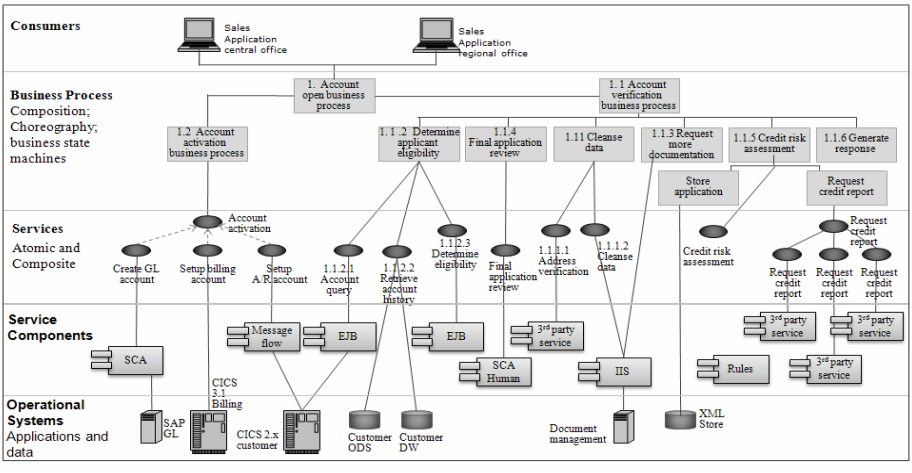


Fig. 1. Account Open SOA Solution Stack

Figure 1 provides a quick overview of the solution stack for the SOA solution. Identified services were utilized through the business process layer. Some services used the indirect exposure realization pattern while other services connected directly to backend systems. The solution stack does not reflect the physical architecture of the solution. The physical architecture of the solution included a layer for the ESB to virtualize access to services.

Table 3 provides an overview of the services used for the Account Open SOA solution. The Interaction and Collaboration (I&C) iteration was the most complex due to the total number of services used within the iteration and the utilized service realization

patterns. The (I&C) iteration's SRP metric value is equal to 1 which is the highest value for the metric. This indicates higher complexity levels for service development since services are using an additional indirection layer to complete their capabilities. On the other hand, flexibility and agility index is higher for the same indirection reasons mentioned earlier due to the loose coupling that indirect exposure injects into a SOA solution.

Table 3. Services and metrics from the Open Account case study

Scenario Service Creation	Service Name	WSIC	NOS 0.83	 0.17	SCP 0.17	DSSS 0.17	SRP 0.5	NHT 0
	Billing Account Setup Service	1	Internal	Internal	Atomic	Static	DE	Automated
	AR Account Inquiry Service	1	Internal	Internal	Atomic	Static	IE	Automated
	AR Account Setup Service	1	Internal	Internal	Atomic	Static	IE	Automated
	Determine Applicant Eligibility Service	1	Internal	Internal	CSwf	Static	IE	Automated
	Credit Report Request Service	1	Internal	Internal	Atomic	Static	DE	Automated
	Address Verification Service	1	External	External	Atomic	Dynamic	DE	Automated
Interaction and Collaboration			**1**	**0**	**0**	**0**	**1**	**0.11**
	Customer Data Cleanse	2	Internal	Internal	Atomic	Static	IE	Automated
	Get Customer Selection of Account	1	Internal	Internal	Atomic	Static	IE	Automated
	Account Rep Review Account Service	1	Internal	Internal	Atomic	Static	IE	Automated
	Message Customer Waiting for Notification	1	Internal	Internal	Atomic	Static	IE	Automated
	Request More Documentation Service	1	Internal	Internal	Atomic	Static	IE	Automated
	Complete Documentation Request Service	1	Internal	Internal	Atomic	Static	IE	Automated
	Final Application Review Service	1	N/A	N/A	N/A	N/A	N/A	Human
	Notify Customer with Decline	1	Internal	Internal	Atomic	Static	IE	Automated
	Notify Customer with Acceptance	1	Internal	Internal	Atomic	Static	IE	Automated
Service Connectivity			**1**	**0**	**0.2**	**0**	**0.8**	**0**
	Setup Billing Account Service	1	Internal	Internal	Atomic	Static	DE	Automated
	Account Activation	4	Internal	Internal	CSs	Static	IE	Automated
	Inquiry AR Customer Account Service	1	Internal	Internal	Atomic	Static	IE	Automated
	Setup AR Account Service	1	Internal	Internal	Atomic	Static	IE	Automated
	Create GL Account Service	2	Internal	Internal	Atomic	Static	IE	Automated

Notes:
- All services were stateless, provided no support for transactions, had only a single version, and utilized virtualized access. These columns are therefore not shown in the table.
- The NOS metric column contains two values since the metric distinguishes between the ratios of external services vs. internal services relative to the overall number of services.

The I&C iteration produced less flexibility due to the existence of a manual human task, however this was seen as a necessary tradeoff because a human needed to verify the final steps of the account for a certain percentage of applicants. This is compensated for by the SRP metric since all I&C iteration service realization patterns are done through indirect exposure which provides looser coupling for the overall SOA solution.

The determination of health of the SOA application based on the short analysis that was completed is a more complex question given the limited amount of data. As indicated earlier in this paper, additional SOA governance parameters need to be evaluated. However, from the limited information that we collected for this case study, there are no red flags that can be raised to indicate an unhealthy behavior or any major issues with the health of the given SOA solution.

6 Future Work

The primary contribution of this paper is the proposed set of metrics that should be tracked and measured for every SOA engagement so that better insight can be gleaned into the complexity, agility and flexibility of the SOA application. However, one of the common problems with new metrics suite is the difficulty of empirically validating and calibrating them across a number of projects. For example, the case study demonstrates

the need to weight each metric to provide more accurate complexity, flexibility, and agility measurements. For example, table 3 shows a slightly higher value of SCP for the Service Connectivity iteration. It also shows a positive value for DSSS metrics which tends to increase complexity. By definition, this should have resulted in more complexity for the Service Connectivity which is true. However, from real life experience the Service Connectivity iteration took longer to accomplish due to increased complexity as a result of other metrics in the table such as SRP and WSIC. Despite these problems, this paper has proposed a reasonable set of metrics, identified as a result of observing numerous SOA deployments. Future work will include empirical assessment of these metrics and the identification of additional ones in order to build a demonstrably useful set of SOA metrics for predicting complexity, flexibility, and agility across a broad spectrum of SOA applications. Additional work is also required to provide clear methods for calculating the values of the proposed aggregate SOA indices and interpret their implications in terms of SOA flexibility, agility and complexity.

Acknowledgments. We would like to thank our colleague Russell Hartmann from IBM for his help in providing the data for the case study.

References

1. Rud, D., Schmietendorf, A., Dumke, R.R.: Product Metrics for Service-Oriented Infrastructure. In: International Workshop on Software Measurement/Metrikon 2006 (2006)
2. Liu, Y.M., Traore, I.: Complexity Measures for Secure Service-Oriented Software Architectures. In: The International Workshop on Predictor Models in Software Engineering (PROMISE 2007) (2007)
3. Qian, K., Liu, J., Tsui, F.: Decoupling Metrics for Services Composition. In: Proceedings of the 5th IEEE/ACIS International Conference and Information Sciences and 1st IEEE/ACIS International Workshop on Component-Based Software Engineering, Software Architecture and Reuse (ICIS-COMSAR 2006) (2006)
4. Arsanjani, A., Allam, A.: Service-Orineted Modeling and Architecture for Realization of an SOA. In: IEEE International Conference on Services Computing (SCC 2006) (2006)
5. Arsanjani, A., Zhang, L., Ellis, M., Allam, A., Channabasavaiah, K.: S3: A Service-Oriented Reference Architecture. IT Pro., 10–17 (June 2007)
6. Chidamber, S.R., Kemerer, C.F.: A Metrics Suite for Object Oriented Design. IEEE Transactions on Software Engineering 20(6), 476–493 (1994)
7. Arsanjani, A., Ghosh, S., Allam, A., Abdollah, T., Ganapathy, S., Holley, K.: SOMA: A method for developing service-oriented solutions. IBM Systems Journal 47(3), 377–396 (2008)
8. Web Services Resource Framework (WSRF), http://docs.oasis-open.org/wsrf/wsrf-primer-1.2-primer-cd-02.pdf
9. Newcomer, E., Lomow, G.: Understanding SOA with Web Services. Addison Wesley, San Francisco (2005)
10. OASIS Web Services Transaction (WS-TX) landing page, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ws-tx
11. Ferguson, D.F., Stockton, M.L.: Service Oriented Architecture: Programming Model and Product Architecture. IBM Systems Journal 44(4), 753–780 (2005)
12. Web Services Context Specification (WS-Context), http://xml.coverpages.org/WS-ContextCD-9904.pdf

Simulation of IT Service Processes with Petri-Nets

Christian Bartsch[1], Marco Mevius[1], and Andreas Oberweis[2]

[1] FZI Research Center for Information Technology, Software Engineering,
Haid-und-Neu-Str.10-14, 76131 Karlsruhe, Germany
{bartsch,mevius}@fzi.de
[2] Universität Karlsruhe (TH), Institute AIFB, Kaiserstraße 89, 76133 Karlsruhe, Germany
{andreas.oberweis}@kit.edu

Abstract. Due to a steadily increasing market for IT services, providers need to set apart from their competitors in order to successfully assert with their service offerings in the market. As a result the improvement of the quality of service process provisioning is becoming a major aspect. In this paper we propose a Petri-net based approach in order to model and simulate service processes in terms of availability levels. Supported by a tool service providers can perform a priori estimations during design time on the potential impact and interaction of availabilities of services involved in provisioning processes. We additionally show how obtained results can be taken into account for negotiating availability levels in Service Level Agreements (SLA).

Keywords: IT Service Process, Availability Pattern, Petri-Net, Process Modeling, Process Simulation.

1 Introduction

Service providers need to set apart from their competitors in order to successfully compete with their service offerings in the market. Besides classical selling propositions such as price, customer proximity or product quality, the quality of providing complete and flexible service processes is becoming a key differentiator from the competition [1]. This is motivated by the fact that the basic thing that matters to the customer is the usage of services based on agreed typical indicators within the domain of IT service management. This could be service availability, time to restore a service, response time or performance level. Furthermore the provisioning of IT services is increasingly based on the modularization of whole service processes. This characteristic offers certain potential for reducing costs and enhancing service quality at the same time [2].

Hidden from customers to a certain extent providers have to establish methods and procedures in order to manage service processes concerning quality aspects. One aspect could be the maximum service availability which can be provided to a customer. This knowledge is important as service level agreements (SLA) between respective stakeholders contain such metrics. Additional penalties might also be taken into account for being part of an SLA in case of nonfeasance. In practice the determination of service levels for availability during the SLA negotiation phase is often based on a

G. Feuerlicht and W. Lamersdorf (Eds.): ICSOC 2008, LNCS 5472, pp. 53–65, 2009.

rule of thumb. Providers offer a certain percentage for a complete service process not knowing about availabilities of partial services involved. A validation of availability values agreed in an SLA usually occurs after the signing of the contract. An a priori estimation for example based on simulation in order to reduce risks of nonfeasance can help providers with identifying feasible levels for service processes. They can "play around" with availability levels of services being part of an offered service process. The precise modeling of service processes and the resources [3] enabling certain process steps utilizing Petri-nets is supposed to support service providers to make assumptions about what availability levels are feasible for potential service offerings. The formal representation allows a systematic a priori simulation of availability values for IT service processes. In addition to this the quality of service processes is affected twofold. On the one hand side service providers are able to identify potential bottlenecks regarding the overall process availability by simulating the impact and interactions of all availabilities of IT services involved. On the other hand the ability to perform a priori estimations could support customers in identifying the required service levels in order to leverage and maximize the performance of their respective business processes. In this paper we introduce a new Petri-net based approach for modeling and simulating service processes in terms of availability levels in order to foster a priori estimations during design time. The approach to be introduced is based on High-level Petri-nets. In High-level Petri-nets, complex objects can be represented. IT service processes are modeled as a manipulation of these objects. High-level Petri-nets not only feature significant advantages with respect to IT service process modeling. Their precise foundation allows straightforward simulation and further extensive analysis. Moreover, High-level Petri-nets support the development and implementation of control and monitoring tools for a continuous improvement of the relevant IT service processes. It is highlighted how Petri-nets can be deployed for service modeling within the IT service management domain in order to improve quality aspects for potentially offered and provided service processes.

In Chapter 2 we discuss related work. We then introduce the basic concepts of Petri-nets and IT service processes in Chapter 3. Afterwards we present in Chapter 4 an approach for modeling the availability of IT services and show how IT service processes can be simulated in order to support contracting service level agreements. Subsequently we present in Chapter 5 two extensive simulation experiments and the findings which can be derived from the results of the simulations.

2 Related Work

Various publications such as [4, 5] contribute to the topic of business process management. Modern languages for business process management (e.g. the Business Process Execution Language (BPEL)[1], Business Process Modeling Language (BPML)[2] or ebXML Business Process Specification Schema (BPSS)[3]) focus on the tracking and execution of business processes by a business application. The formal analysis and monitoring of process performance is not considered. Different methods

[1] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel
[2] http://www.omg.org/docs/dtc/06-02-01.pdf
[3] http://www.ebxml.org/

and description languages have been proposed for process modeling. Most of them are based on textual programming languages or graphical notations such as dataflow diagrams, UML state charts, Event-driven Process Chains (EPCs), Petri-nets or related notations [6, 7]. They can be divided into semi-formal methods, e.g., EPCs [8], and formal methods, e.g., Petri-nets [9, 10]. Semi-formal methods, especially EPCs, are very popular, widely disseminated and easy to use. They do, however, exhibit major shortcomings especially if they support the modeling and simulation of business processes with one integrated method [11]. These shortfalls can be overcome by the use of Petri-nets. On the basis of different definitions and interpretations of their static and dynamic modeling components different types of Petri-nets have been derived to date. High-level Petri-nets (Predicate/Transition nets [12] or Colored Petri-nets [13]) have proven to be suitable for modeling complex dynamic systems. They support an integrated description of structural and behavioral characteristics of dynamic systems. In recent years, enhancements have been made in order to improve the modeling of objects with more complex structures using High-level Petri-nets. Nested Relation/Transition nets combine the concept of complex structured objects with the Petri-net formalism [14]. Motivated by the increasing acceptance and dissemination of the XML standard [15], XML-nets were derived (for a detailed description we refer to [16]). The major advantages of the process-oriented High-level Petri-net-based modeling of IT service processes can be summarized as follows:

- High-level Petri-nets are capable of modeling and simulating availability levels of IT services in order to foster a priori estimations during design time.
- High-level Petri-nets provide capabilities for modeling complex objects with a hierarchical structure which can typically be found in IT service processes.
- By using standards, High-level Petri-nets support the intra- and inter-organizational exchange of service level information and thereby foster standardized design, implementation and monitoring of pre-defined performance levels.
- A High-level Petri-nets-based software prototype can be directly linked to simulation relevant information. It automatically simulates and executes pre-defined simulation runs of IT service processes.

For these reasons, High-level Petri-nets are not only suited for modeling business processes but also for the integrated process simulation and analysis especially in the domain of IT services and Service Level Management. SLM comprises proactive methodology and procedures to ensure that adequate availability levels of IT services are delivered to all IT users in accordance with business priorities and at acceptable cost [17]. One of the main ways for publishing services is by setting up a service level agreement (SLA) between the IT department and the business. The SLA is a document that defines and identifies the various services offered by a service provider to its clients. Since SLAs provide the term and conditions that define the services rendered by a provider, it then limits its scope of support, therefore minimizing production cost [18, 19]. It describes the services along with a regime for managing how well these are delivered. They are typically defined in terms of the availability, responsiveness, integrity and security delivered to the users of the service [17]. It states a commitment to range, quantity and quality of services of service providers. The size of service levels must be justifiable and benefit the business as a whole. If this is not the case then such service levels should be renegotiated [20]. Various research

approaches and results exist in the context of SLAs and quality of service (QoS) within respective domains such as Web Services [21, 22, 23], mobile communication [24], IT Management [25, 26] or supply chain management [27]. They primarily focus on the modeling, monitoring and analysis of SLAs but scarcely use the benefit of simulation for determining the impact of IT services on the service processes they support. This knowledge could be used – especially if considering service availability aspects – in order to enhance the definition of service levels during design time.

3 Petri-Nets and IT Service Processes

Petri-nets [10] are a formal graphical process description language that combines the advantages of the graphical representation of processes with a formal definition. They allow the representation of a class of real-world processes by utilizing a graphical process schema. Petri-nets consist of static components (places, depicted by circles) and dynamic components (transitions, depicted by rectangles) that are connected by edges. In the following we assume that the reader has basic knowledge about the functionality of Petri-nets. For illustration purposes we present a Place/Transition net in Figure 1 that describes a simplified example derived from the IT Service Management domain.

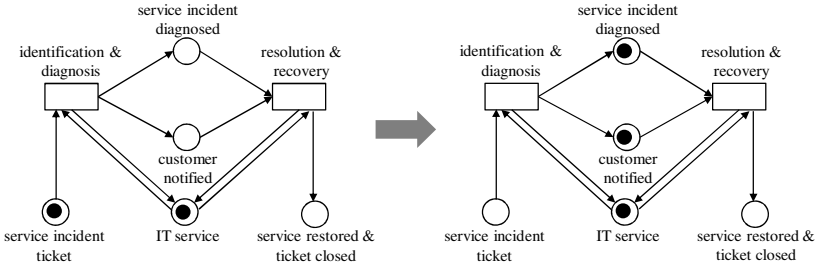


Fig. 1. Example for Place/Transition net

A ticket, represented as an individual object in place "service incident ticket", can be processed by transition "identification&diagnosis" if an object is available in place "IT service". The transition can fire and because of a branch the ticket object is fired to the places "service incident diagnosed" and "customer notified". As there is a loop between place "IT service" and transition "identification&diagnosis" the same object is sent back to its origin place.

The major task of business performance analysis on the basis of availability levels is the assessment of alternative business process designs with respect to a given set of objectives. In addition to methods commonly applied in Business Process Management (BPM), in particular High-level Petri-nets support analysis based on simulation. Simulation in process-oriented management of IT service processes aims, for example, at validating the defined exceptional states and the corresponding customizing activities. Due to their formal foundation Petri-nets can directly be executed and therefore be used to simulate the described process [28]. A simulation engine

interprets the formal syntax and semantics of the Petri-net and transforms it into a machine-readable code [6]. A widespread technique to formally model services, is based on Petri-nets [3].

4 Availability Modeling and Simulation of IT Service Processes

In this section we present an approach to model and configure the availability of IT services and the service processes they support. The availability of an IT service is the percentage of possible time that the service is online or available to use. A service should be available on demand as often as possible in order to be able to serve users needs when they need them to be served [29]. We assume that in case of an IT service enabling the execution of certain activities within service processes it is mandatory that a service is available as soon as the processing of a certain object through an activity requires it. Figure 2 illustrates the states of service availability we differentiate within the IT service management domain.

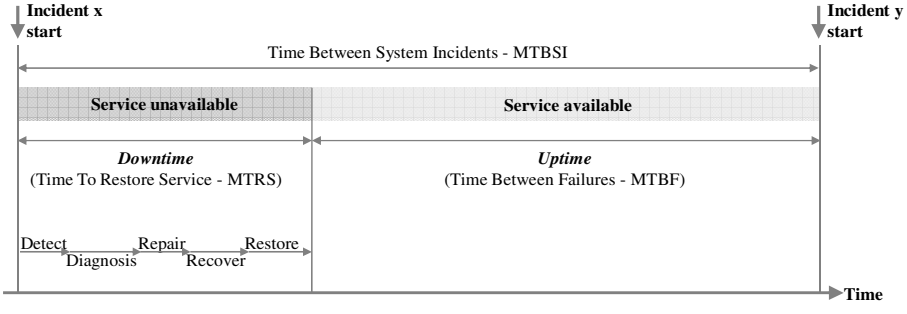


Fig. 2. States of service availability

As soon as any internal or external event triggers an incident x of an IT service and is detected by respective tools it passes into the state *unavailable*. The total period that an IT service is unavailable for operational use within an agreed service time is called *downtime* $dt(s_n)$.

$$dt(s_n) = (1 - a(s_n)) * ast(s_n) \quad (1)$$

In order to finally restore the failed IT service several steps such as diagnosing the cause of fault as well as repairing and recovering the service must be processed. All steps can take a different time. The period from detecting to restoring the IT service is the *time to restore a service* (MTRS). The service is now available again.

The determination of service availability always requires an *agreed service time* $ast(s_n)$ being the value during which a particular IT service s_n is agreed to be fully available. The total period that an IT service is operational within an agreed service time is called *uptime* and takes the probability $a(s_n)$ into account – being identical with key figure a*vailability* – to which a service will not fail.

$$ut(s_n) = a(s_n) * ast(s_n) \quad (2)$$

The time until the next incident y causes another failure of the same IT service is the *time between two failures* (MTBF). The *Availability* $a(s_n)$ of an IT service is the ability to perform its required function over a stated period of time. The availability of a service considers all occurred downtimes during an agreed service time. We define:

$$a(s_n) = \frac{ast(s_n) - \sum dt(s_n)}{ast(s_n)} \tag{3}$$

Availability pattern for IT services

For simulation purposes the formal modeling of service processes and the respective IT service availabilities with Petri-nets requires closer examination on the representation of service downtimes and the corresponding *Time To Restore Service* $MTRS(s_n)$ involved. With the following pattern shown in Figure 3 we can model the aforementioned metrics.

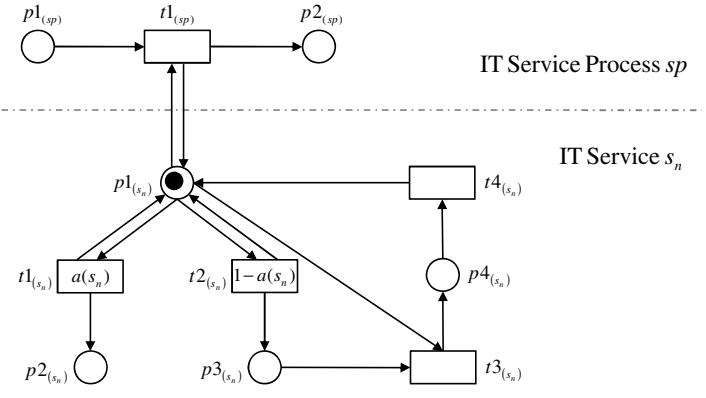


Fig. 3. Availability pattern for IT services

It can therefore be interpreted as representative of a respective IT service s_n in terms of availability. The proposed availability pattern based on Petri-nets can be used as a type of resource in order to enable the execution of a (semi-) automatic process step within a service process. A transition $t_m(sp)$ is connected to the respective supporting IT service s_n. A marking M of the considered IT service process sp and IT services s_n enables a transition $t_m(sp)$ if it marks every place in $\bullet t_m(sp)$. If $t_m(sp)$ is enabled within M, then it can fire, and its firing leads to the successor marking M' ($M \xrightarrow{t_m(sp)} M'$). In this depiction we simplified the IT service process and assume a linear process executing process steps sequentially. For simulation purposes each pattern – representing a single IT service s_n – needs an individual initial configuration in terms of probability of service failure $1 - a(s_n)$ and the average period from detecting an IT service failure to restoring the service for operational $MTRS_{sim}(s_n)$. Especially the last metric, represented by transition $t3(s_n)$, requires closer examination as its value is comprised of further data. The *checking interval* $ci(s_n)$ is a defined repetitive timing after which transitions $t1(s_n)$ and $t2(s_n)$ is enabled to fire an object.

The introduction of transition $t4(s_n)$ is supposed to illustrate the option for integrating additional activities such as accounting, reporting, penalty handling etc.

The checking cycle $cc(s_n)$ is a supporting construct in order to determine the total number of potential failures within an agreed service time. Due to the structure of the availability pattern we use uptime $ut(s_n)$ for further calculation instead of $ast(s_n)$. Transitions $t1(s_n)$ and $t2(s_n)$ can only fire – and therefore check for availability – if $p1(s_n)$ is marked. This means that the IT service is up and running.

$$cc(s_n) = \frac{ut(s_n)}{ci(s_n)*100} \tag{4}$$

Running a cycle equates to a hundred proceeded availability checks. This means for example that for a given service availability of 98.0 percent transition $t1(s_n)$ fires 98 objects (= ping for availability) to $p2(s_n)$ and $t2(s_n)$ 2 objects (= service failure) to $p3(s_n)$ on average. In order to calculate $dtast(s_n)$, the total amount of "Downtime objects" in $p3(s_n)$ per $ast(s_n)$ as shown in (6), we need to identify the number of potential *downtimes per checking cycle* $dtcc(s_n)$ at first.

$$dtcc(s_n) = (1 - a(s_n)) * 100 \tag{5}$$

$$dtast(s_n) = dtcc(s_n) * cc(s_n) \tag{6}$$

The value for $MTRS_{sim}(s_n)$ which is represented in the pattern by transition $t3(s_n)$ allows the modeling of the maximal allowed downtime per agreed service time. We calculate the minimum as shown in (7.1) because the value of $MTRS_{sim}(s_n)$ can't be higher than the maximum downtime derived from the given service availability.

$$MTRS_{sim}(s_n) = \min\left(dt(s_n), \frac{dt(s_n)}{dtast(s_n)}\right) \tag{7.1}$$

On closer examination of $MTRS_{sim}(s_n)$ it can be seen that the value is individual for every given service availability $a(s_n)$ considering respective values for $ast(s_n)$ and $ci(s_n)$ which leads us to following:

Thesis.

$$\frac{dt(s_n)}{dtast(s_n)} = \frac{ci(s_n)}{a(s_n)}$$

Proof.

$$\frac{dt(s_n)}{dtast(s_n)} = \frac{(1 - a(s_n)) * ast(s_n)}{dtcc(s_n) * cc(s_n)} = \frac{(1 - a(s_n)) * ast(s_n)}{(1 - a(s_n)) * 100 \ * \frac{ut(s_n)}{ci(s_n) * 100}}$$

$$= \frac{(1 - a(s_n)) * ast(s_n)}{\frac{(1 - a(s_n)) \ * ut(s_n)}{ci(s_n)}} =$$

$$\frac{ci(s_n) * ast(s_n)}{ut(s_n)} = \frac{ci(s_n) * ast(s_n)}{a(s_n) * ast(s_n)} = \frac{ci(s_n)}{a(s_n)}$$

As a result we can define (7.2):

$$MTRS'_{sim}(s_n) = \min\left(dt(s_n), \frac{ci(s_n)}{a(s_n)}\right) \tag{7.2}$$

This implication eases the identification of values for $MTRS'_{sim}(s_n)$ because the checking interval $ci(s_n)$ as well as service availability $a(s_n)$ serve as direct input into the model and don't need to be calculated separately compared to (7.1).

Representation of IT service processes

For our purposes we model service processes using the duration d of a single process step $sp_{l,x}$ and the IT services enabling respective step. The service process is depicted as a sequence of tupels $[d(sp,x); a(s_n)]$, where $d(sp,x)$ is the duration of process step x within a service process sp and $a(s_n)$ indicates the availability of IT services s_n involved in the execution of (sp,x). Exemplarily an IT service process A containing three process steps and supported by two IT services could then be coded as: $A = [d(A,1); a(s_1)] \rightarrow [d(A,2); a(s_1), a(s_2)] \rightarrow [d(A,3)]$.

$(A,1)$ is supported by s_1 and the respective service availability $a(s_1)$ whereas $sp_{A,3}$ is not supported by an IT service at all. In order to finally gain the overall availability of a service process $a(sp)$ we first need to identify the maximum number of objects the service process can go through if IT services involved provide a hundred percent availability. The number of objects in the end place of the service process modeled with Petri-nets completely depends on the duration of each process step and therefore defines an upper bound not being constrained by any IT service availability. We then determine the impact of single IT service availability to the overall availability of the supported IT service process by comparing the number of objects in the end place after a simulation period in relation to the predetermined upper bound.

5 Evaluation

In Figure 4 we present a simplified use case derived from a conducted IT service management project[4].

For this simulation example we assume that all IT services perform independent from each other. The checking interval within each IT service was set to 60 minutes. The scenario includes three IT services supporting two incident management processes with different execution times. It basically differs in the number of IT services involved. As the example shows, an IT service is available as long as an object lies within a place $p1(s_n)$light turns into a green state or red state as soon as a. The availability pattern, as depicted in Figure 3, was modeled with a simulation tool being under ongoing development[5].

As soon as parallel and alternative paths will become part of the simulation complexity will increase noticeably. As a consequence more simulation cycles will be necessary in order to keep significance of the simulation results.

[4] Please note that other service process configurations containing alternatives and parallelism would also be possible but are not mentioned in this example.

[5] A deployable version of the simulation tool will be open to the public soon.

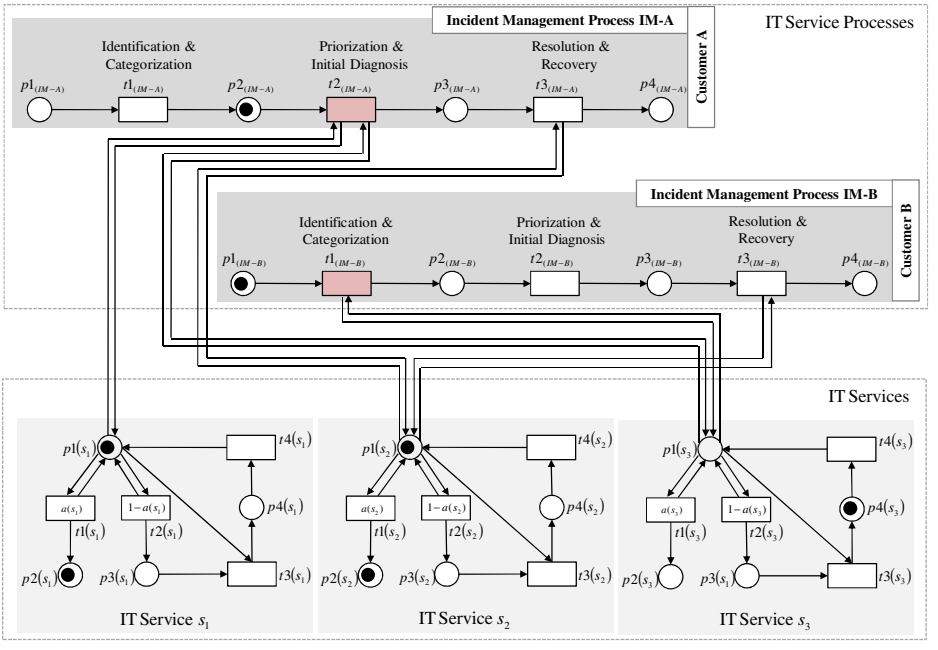


Fig. 4. IT service process availability simulation

Simulation Scenario A: 3 process steps | 3 IT services (2+1)
The first scenario as shown in Figure 5(a) simulates a service process $IM - A$ supported by IT services s_1, s_2 and s_3 whereas $(IM - A, 2)$ is supported by two services s_2 and s_3 simultaneously. The configuration for simulations $SimA1 - SimA5$ is as follows:

$$IM - A = [d(IM - A, 1) = 3] \rightarrow [d(IM - A, 2) = 5; a(s_1), a(s_3)] \rightarrow [d(IM - A, 3) = 10; a(s_2)]; \quad \mathrm{d}(IM - A) = 18;$$

The parameters for $a(s_1), a(s_2)$ and $a(s_3)$ – represented by dark grey bars in the figure – are changed for each simulation in order to find out the availability of the service process – represented by the light grey bars – at the end of simulation period of 60 days. Results show that $a(IM - A)$ of $IM - A$ is rather proportional to $a(s_n)$. A service provider must therefore increase the availability of selected IT services in order to increase the availability of the service process. The demand for high process availability usually implies a high (monetary) effort to ensure high IT service availability.

If we assume that the duration of service process steps themselves plays a role for respective service process availability we reconfigure sp_{IM-A} to:

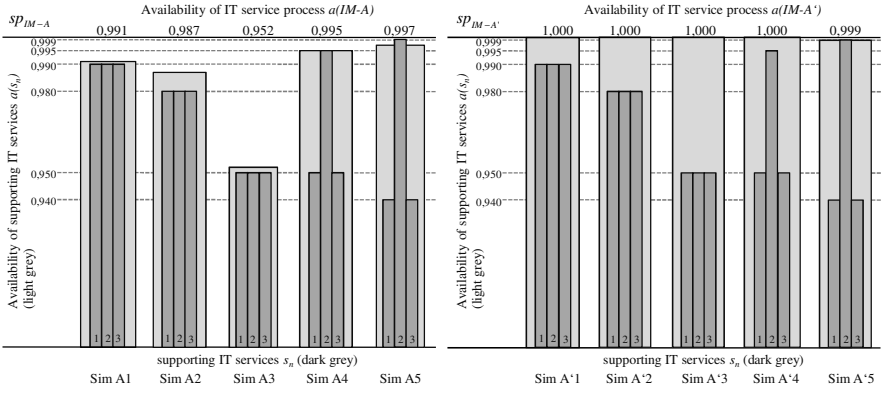
$$IM - A' = [d(IM - A', 1) = 3] \rightarrow [d(IM - A', 2) = 8; a(s_1), a(s_3)] \rightarrow [d(IM - A', 3) = 7; a(s_2)]; \quad d(IM - A) = 18;$$

We notice, that slight changes in $d(IM - A', 2)$ and $d(IM - A', 3)$ cause the effect that $a(IM - A)$ stays on a continuous high level of a hundred percent availability for all simulations $SimA'1 - SimA'5$. As a consequence this means that for service

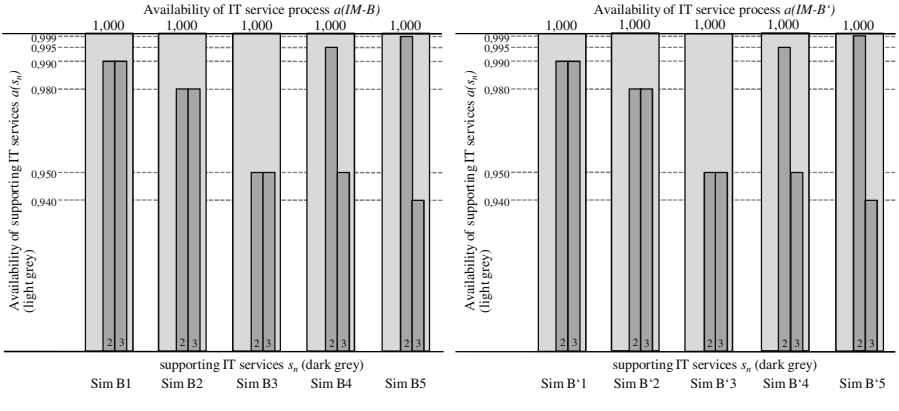
process $sp_{IM-A'}$ an availability level for the supporting IT services $a(s_1), a(s_2)$ and $a(s_3)$ of 95,00% would be sufficient in order to enable an availability of the supported IT service process of 100,00%. This finding is helpful for negotiating SLAs as there is no difference to the overall performance of the service process if paying for a 99,90% or for a 95,00% IT service availability and can save a customer money.

Simulation Scenario B: 3 process steps | 2 IT services (1+1)
As we can see from figure 5(b), simulations $SimB1 - SimB5$ as well as $SimB'1 - SimB'5$ for scenario B show the same results although we changed the duration of $(IM - B, 2)$ and $(IM - B, 3)$ in $IM - B$:

$$IM - B = [d(IM - B, 1) = 3; a(s_4)] \rightarrow [d(IM - B, 2) = 5] \rightarrow [d(IM - B, 3) = 10; a(s_2)]; \quad d(IM - B) = 18;$$
$$IM - B' = [d(IM - B', 1) = 3; a(s_4)] \rightarrow [d(IM - B', 2) = 3] \rightarrow [d(IM - B', 3) = 8; a(s_2)]; \quad d(IM - B') = 14;$$



(a)



(b)

Fig. 5. (a) Simulation results for service processes $\boldsymbol{IM - A}$ and $\boldsymbol{IM - A'}$. (b) Simulation results for service processes $\boldsymbol{IM - B}$ and $\boldsymbol{IM - B'}$.

Different availability levels for supporting IT services don't have any influence to the maximum service process availability of 100,00%. As in Scenario A 95,00% IT service availability would be sufficient in order to support the process best. The additional finding is that, even though we changed the duration of process steps all availabilities remained the same. This could be a hint of a much more *stable* service process $IM - B$ compared with $IM - A$ as slight changes in the duration in combination with different IT service availabilities don't affect the overall process performance at all.

6 Conclusion and Outlook

In this paper we presented a simulation-based approach in order to identify potential impact and interdependencies in terms of availability between IT services and the service process supported. This can assist service providers and their customers when negotiating SLAs during design time. The stakeholders involved now have a possibility to simulate a more realistic service availability level which can support their processes at most. The current approach can't handle the effect that as soon as an object is processed by a process step (transition) and the supporting IT service (place) fails along the way the affected object should actually be considered lost. At the moment the simulation only considers the incoming moment for a transition to be locked or active. As soon as an object "enters" a transition, it will be processed in any case. We need further research on complexity and performance aspects of current simulations as they are very resource intensive already, even though the examples are quite simple. We are working on an extension of modeling and simulating IT services by utilizing XML-nets [16] in order to exemplarily include rules and additional information about an object to cope with the current constraint. The objective is the Petri-net based simulation of whole SLAs described as XML-Documents.

References

1. Bartsch, C., Shwartz, L., Ward, C., Grabarnik, G., Buco, M.J.: Decomposition of IT service processes and identification of alternatives using ontogoly. In: IEEE/IFIP Network Operations and Management Symposium (NOMS), pp. 714–717. IEEE Computer Society Press, Los Alamitos (2008)
2. Böhmann, T., Junginger, M., Krcmar, H.: Modular Service Architectures - A Concept and Method for Engineering IT Services. In: Proceedings of the 36th Annual Hawaii International Conference on System Sciences, pp. 74–83. IEEE Computer Society Press, Los Alamitos (2003)
3. Reisig, W.: Towards a Theory of services. In: Kaschek, R., et al. (eds.) UNISCON 2008. LNBIP, vol. 5, pp. 271–281. Springer, Heidelberg (1974)
4. van der Aalst, W.M.P., Desel, J., Oberweis, A. (eds.): Business Process Management. LNCS, vol. 1806. Springer, Heidelberg (2000)
5. van der Aalst, W.M.P., ter Hofstede, A.H.M., Weske, M.: Business Process Management: A Survey. In: van der Aalst, W.M.P., ter Hofstede, A.H.M., Weske, M. (eds.) BPM 2003. LNCS, vol. 2678, pp. 1–12. Springer, Heidelberg (2003)
6. Desel, J., Erwin, T.: Modeling, Simulation and Analysis of Business Processes. In: van der Aalst, W.M.P., Desel, J., Oberweis, A. (eds.) Business Process Management. LNCS, vol. 1806, pp. 129–141. Springer, Heidelberg (2000)

7. Karagiannis, D., Juninger, S., Strobl, R.: Introduction to Business Process Management Systems Concepts. In: Scholz-Rieter, S. (ed.) Business Process modeling, pp. 81–106 (1996)
8. Scheer, A.-W.: ARIS – Business Process Modeling, 3rd edn. Springer, Berlin (2007)
9. Reisig, W.: Place/Transition Systems. In: Brauer, W., Reisig, W., Rozenberg, G. (eds.) APN 1986. LNCS, vol. 254, pp. 117–141. Springer, Heidelberg (1987)
10. Reisig, W., Rozenberg, G. (eds.): Lectures on Petri Nets I: Basic Models. LNCS, vol. 1491. Springer, Heidelberg (1998)
11. van der Aalst, W.M.P.: Formalization and Verification of Event-driven Process Chains. Information and Software Technology 41(10), 639–650 (1999)
12. Genrich, H.J.: Predicate/Transition Nets. In: Brauer, W., Reisig, W., Rozenberg, G. (eds.) Petri Nets: Central Models and Their Properties, Advances in Petri Nets, pp. 207–247. Springer, Heidelberg (1986)
13. Jensen, K.: Coloured Petri Nets. In: EATCS Monographs on Theoretical Computer Science. Basic Concepts, vol. 1. Springer, Berlin (1992)
14. Oberweis, A.: An integrated approach for the specification of processes and related complex structured objects in business applications. Decision Support Systems 17(1), 31–53 (1996)
15. Bray, T., Paoli, J., Sperberg-McQueen, C.M., Maler, E., Yergeau, F., Cowan, J.: Extensible Markup Language (XML) 1.1 (Second Edition). W3C Recommendation 16, edited in place 29, World Wide Web Committee (W3C) (August 2006)
16. Lenz, K., Oberweis, A.: Inter-organizational Business Process Management with XML Nets. In: Ehrig, H., Reisig, W., Rozenberg, G., Weber, H. (eds.) Petri Net Technology for Communication-Based Systems. LNCS, vol. 2472, pp. 243–263. Springer, Heidelberg (2003)
17. Sturm, R., Morris, W., Jander, M.: Foundations of Service Level Management. Sams (2000)
18. Blokdijk, G.: Service Level Agreement 100 Success Secrets. Lightning Source UK Ltd. (2008)
19. Johnston, R., Clark, G.: Service Operations Management – Improving Service Delivery. Pearson Education Limited, Essex (2005)
20. Bruton, N.: Managing the IT services process. Butterworth-Heinemann, Burlington (2004)
21. Ludwig, H.: Web Services QoS: External SLAs and Internal Policies – Or: How do we deliver what we promise? In: Proceedings of the 4th IEEE International Conference on Web Information Systems Engineering Workshops, pp. 115–120. IEEE CS Press, Los Alamitos (2003)
22. Hudert, S., Ludwig, H., Wirtz, G.: Negotiating SLAs - An approach for a generic negotiation framework for WS-Agreement. In: Proceedings of the 20th International Conference on Software Engineering & Knowledge Engineering (SEKE 2008), pp. 587–592 (2008)
23. Wohlstadter, E., Tai, S., Mikalsen, T., Rouvellou, I., Devanbu, P.: GlueQoS: Middleware to Sweeten Quality-of-Service Policy Interactions. In: Proceedings of the 26th International Conference on Software Engineering (ICSE 2004), pp. 189–199. IEEE Computer Society, Los Alamitos (2004)
24. Song, M., Chang, R., Song, R., Song, J.: Research on the SLA-based Service Management in Mobile Communication Network. In: Canadian Conference on Electrical and Computer Engineering (CCECE 2004), vol. 2, pp. 1017–1020 (2004)

25. Anders, T.: Development of a generic IT service catalog as pre-arrangement for Service Level Agreements. In: 10th IEEE Conference on Emerging Technologies and Factory Automation (ETFA 2005), vol. 2, pp. 567–573 (2005)
26. Wen-Li, D., Hang, Y., Yu-Bing, Z.: Testing BPEL-based Web Service Composition Using High-level Petri Nets. In: Proceedings of the 10th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2006), pp. 441–444 (2006)
27. Karhunen, H., Eerola, A., Jantti, M.: Improving Service Management in Supply Chains. In: Proceedings of the International Conference on Service Systems and Service Management, vol. 2, pp. 1415–1420 (2006)
28. Mochel, T., Oberweis, A., Sänger, V.: INCOME/STAR: The Petri net simulation concepts. Systems Analysis – Modeling – Simulation. Journal of Modeling and Simulation in Systems Analysis 13(1-2), 21–36 (1993)
29. Addy, R.: Effective IT Service Management. Springer, Heidelberg (2007)

Automatic Test Case Generation for Interacting Services

Kathrin Kaschner and Niels Lohmann

Universität Rostock, Institut für Informatik, 18051 Rostock, Germany
{kathrin.kaschner,niels.lohmann}@uni-rostock.de

Abstract. Service-oriented architectures propose loosely coupled interacting services as building blocks for distributed applications. Since distributed services differ from traditional monolithic software systems, novel testing methods are required. Based on the specification of a service, we introduce an approach to automatically generate test cases for black-box testing to check for conformance between the specification and the implementation of a service whose internal behavior might be confidential. Due to the interacting nature of services this is a nontrivial task.

Keywords: Testing, Interacting Services, Test Case Generation.

1 Introduction

Software systems continuously grow in scale and functionality. Various applications in a variety of different domain interact and are increasingly dependent on each other. Then again, they have to be able to be adjusted to the rapidly changing market conditions. Accordingly complex is the development and changing of these distributed enterprise applications.

To face these new challenges a new paradigm has emerged: *service-oriented computing* (SOC) [1]. It uses *services* as fundamental building blocks for readily-creating flexible and reusable applications within and across organizational boundaries. A service implements an encapsulated, self-contained functionality. Usually, it is not executed in isolation, but interacts with other services through message exchange via a well-defined interface. Thus, a system is composed by a set of logically or geographically distributed services, and SOC replaces a monolithic software system by a composition of services.

The encapsulation of a functionality further supports the reuse of services. To this end, a *service-oriented architecture* (SOA) proposes a service repository managed by a broker which contains information about services offered by several service providers. Due to trade secrets, a service provider will not publish a verbatim copy of his service containing all business and implementation details, but only an abstract description thereof. This description — called *public view* — only contains the information necessary to interact correctly with the actual implemented service.

While literature agrees that a public view or a similar description is necessary to realize an SOA, only few concrete approaches to generate public views have

G. Feuerlicht and W. Lamersdorf (Eds.): ICSOC 2008, LNCS 5472, pp. 66–78, 2009.

been proposed, for instance [2]. Even worse, interaction became more complex as services evolved. Instead of simple remote-procedure calls (i. e., request/response operations), arbitrarily complex and possibly stateful interactions on top of asynchronous message exchange are common [3]. In addition control flow, ownerships of choices, semantics of messages, and nonfunctional properties have to be taken into account. Then again, also the service provider is interested in whether the public view of his service implements the same business protocol as the actual implemented service. That means, every interaction that a customer can derive from the public view should also be valid together with the implemented service — even without knowledge about non-disclosed business and implementation details. This is especially important since the public view is used by a service requester to check whether his service fits to the provider's implemented service.

To this end, verification and testing of services received much attention both in academia and industry. However, the communicating nature of services is often neglected. Instead of taking the new characteristics of SOC and SOA into account, existing work on testing classical software systems was slightly adapted to cope with the new languages used to implement services. Consequently, these approaches only take the internal behavior of a service into account, but ignore the interactions with other services. Others in turn are restricted to stateless remote-procedure calls. This in turn might lead to undetected errors.

In this paper, we suggest a novel approach to test whether an implemented service interacts as it is warranted by its public view. Thereby, we focus on the business protocol (i. e., the message exchange and control flow) of the service. Aspects such as semantics of messages or nonfunctional properties are out of scope of this paper, but are orthogonal to our approach. We claim that like a function call is the most natural test case to test a function of a classical program, a *partner service* is likewise the most natural test case for another service. But as mentioned earlier, a public view implicitly describes a large number of correct interactions, and both finding and testing all possibilities is a tedious task. Therefore, we propose an approach to automatically generate test cases (i. e., partner services) that are required to test whether an implementation conforms to a public view.

The rest of the paper is organized as follows. In the next section, we describe how black-box testing can be realized using interacting services. The main contribution of this paper is presented in Sect. 3 where we explain our test case generation approach and present an algorithm to further minimize the generated test set. In Sect. 4, we focus on related work and highlight the differences to our approach. Section 5 concludes the paper.

2 Testing Interacting Services

Best practices propose that complex systems should be *specified* prior to their implementation. A specification of a service describes its business protocol and already contains all relevant internal decisions (e. g., the criteria whether or not

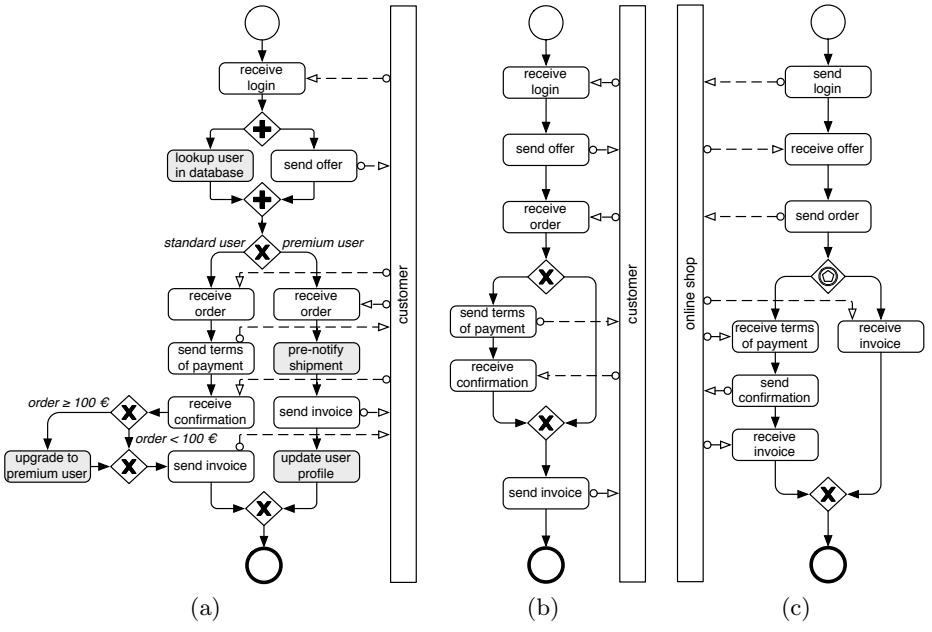


Fig. 1. A specification (a) and a public view (b) of a simple online shop together with a customer service (c) Internal actions are depicted gray

a credit should be approved), but lacks implementation-specific details (e. g., whether amounts are represented as integers or floats). Finally, the public view lacks both details about internal decisions and implementation, and only describes the business protocol that is realized by the services. Ideally, the business protocol defined by the public view, the specification, and the implementation coincides. In particular, if *any* service behavior that is derived from the public view (resp. the specification) is a valid interaction with the implemented service, we call the implementation *conformant* with the public view (resp. the specification).

As the running example of this paper, consider an online shop which processes an order of a customer. Its specification is depicted as a BPMN diagram [4] in Fig. 1(a). After a customer logs in to the online shop, its user name is looked up in the shop's database. Depending on previous orders, the customer is rated as standard or premium customer. On the one hand, standard customers have to confirm the terms of payment and, based on the amount of the ordered goods, can be upgraded to premium users. On the other hand, the shipment to premium users is handled with priority. An actual implementation of this specification (e. g., in BPEL [5]) is straightforward.

To interact correctly with the online shop, certain information does not need to be disclosed and Fig. 1(b) depicts a public view of the online shop. Non-communicating activities (depicted gray in the specification) as well as branching conditions were removed. This public view now describes the online shop's

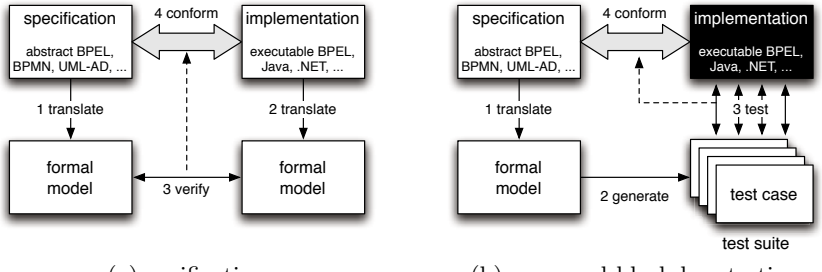
business protocol, but does not disclose unnecessary internal information. A customer can check whether his service (see Fig. 1(c)) fits to this public view. The example shows that sheer mirroring of the public view's message flow is insufficient to achieve deadlock-free interaction, because the customer cannot influence the shop's decision whether to be recognized as standard or premium customer. Therefore, the customer must be ready to both receive the terms of payment *and* the invoice after he sent his order to the shop; in Fig. 1(c), this is expressed by an event-based exclusive gateway.

In the following, we concentrate on checking conformance between a specification and an implementation of a service; that is, we focus on the service provider's point of view who has access to both the specification and the implementation. Nevertheless, the approach is likewise applicable to check conformance between a public view and an implementation of this public view. This might be relevant for service brokers who want to assert conformance between their stored public views and the respective implemented services.

2.1 Verification vs. Testing

A widespread realization of services are *Web services* [6] which use WSDL to describe their interface and SOAP to exchange messages. While usually the specification language (e. g., BPMN, UML activity diagrams, EPCs) differs from the implementation language (e. g., Java, .NET languages), the Web Service Business Process Execution Language (BPEL) can be used to both specify and implement Web services. The specification (called an abstract BPEL process) contains placeholders that are replaced by concrete activities in the implementation (the executable BPEL process). The implementation can thereby be seen as a refinement of the specification.

One possibility to check conformance is *verification*, see Fig 2(a). Thereby, the implementation and its specification have to be translated into a formal model (1, 2). Conformance between the models (3) then implies conformance between specification and implementation (4). Obviously, this approach can only be applied if both models are available.



(a) verification (b) proposed black-box testing

Fig. 2. Approaches to check conformance between specification and implementation

However, there are several scenarios in which a verification is impossible, because the formal model of the implementation is not available. For example, parts or the whole implementation might be subject to trade secrets. Furthermore, programming languages such as Java are — compared to high-level languages like BPEL — rather fine-grained and it is therefore excessively elaborate to create a formal model for such an implementation.

As an alternative, the implementation can be *tested*. The major disadvantage is that by testing only the presence of errors can be detected, but not their absence. To nevertheless be able to make a statement about the correctness of an implementation, a test suite with a significant number of test cases is necessary.

To this end, we translate the specification into a formal model (1). This transformation should be possible, because the specification is usually not as complex as, for example, a Java program. The model of the specification is then the base for generating the test cases (2) with which we test the implementation (3). If a test fails, we can conclude that the implementation does not comply to the specification (4). Because the tester does not need knowledge about the code of the implementation, our approach is a *black-box testing* method.

Since a large number of test cases may be required for an adequate test, they should be automatically generated. This does not only save time and costs during the development, but furthermore a systematic approach can generate test cases which are hard to find manually. These test cases in turn allow for the detection of errors that were missed by manually designed test cases.

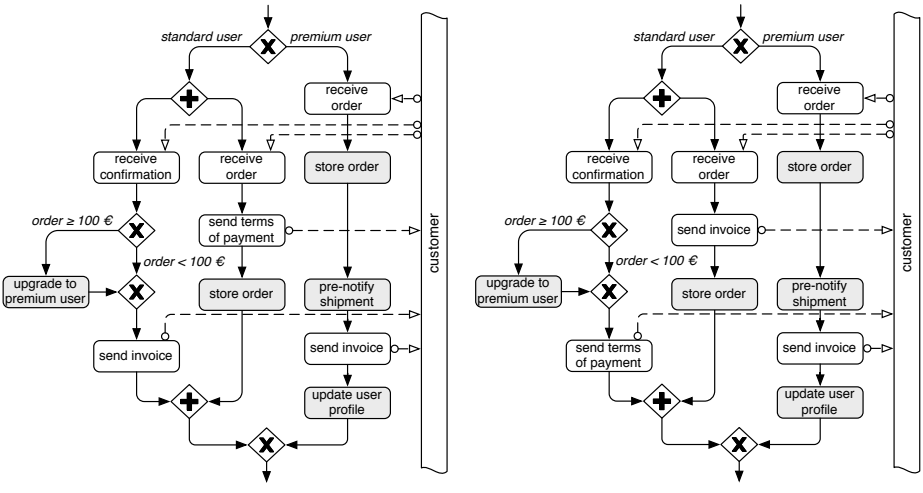
2.2 Testing Services by Partner Services

As motivated in the introduction, we propose to use services to test an implemented service. Considering the example online shop, any partner service of the specification of Fig. 1(a) can serve as one test case to test conformance of an implementation of the online shop. An example for a test case is the customer service depicted in Fig. 1(c). Its composition with the specification is free of deadlocks, and the same should hold for its composition with the implemented online shop.

A general test procedure for interacting services can be sketched as follows. The service under test is deployed in a testing environment together with the test suite. To process a whole test suite, the contained test cases are executed one after the other. Thereby, each test case follows the business protocol derived from the specification and interacts by message exchange with the service under test. The testing environment then is responsible for logging and evaluating exchanged messages. One implementation of such a test environment is the tool BPELUnit [7]. However, in BPELUnit the test suite has to be created manually.

Two implementations for the example online shop are depicted in Fig. 3.[1] Beside insertion of internal activities such as “store order”, also reordering of ac-

[1] To ease the presentation, we again use BPMN to diagram the implementations and focus on the message exchange between online shop and customer. Additionally, only an excerpt is depicted.



(a) conformant implementation (b) incorrect implementation

Fig. 3. Parts of two implementations of the online shop

tivities does not necessarily jeopardize conformance [8]. For example, implementation in Fig. 3(a) receives the order and the confirmation message concurrently, yet still is conformant to the specification in Fig. 1(a). In contrast, the implementation in Fig. 3(b) further exchanges the invoice and the terms of payment message. This implementation does not comply with the specification.

This inconformance can be detected by composing the test case of Fig. 1(c) to the implementation in Fig. 3(b). If the customer is treated as standard customer, the online shop — after receiving the login and an order and sending the offer and the invoice — is left in a state where it awaits the confirmation message. This will, however, only be sent by the customer test case after receiving the terms of payment. The services wait for each other; their composition deadlocks. While this test case can be easily derived directly from the specification, manual test case generation becomes an increasingly tedious task for real-life specifications with thousands of partner services.

The next section explains how test cases such as the customer service of Fig. 1(c) can be automatically generated.

3 Generating Test Cases

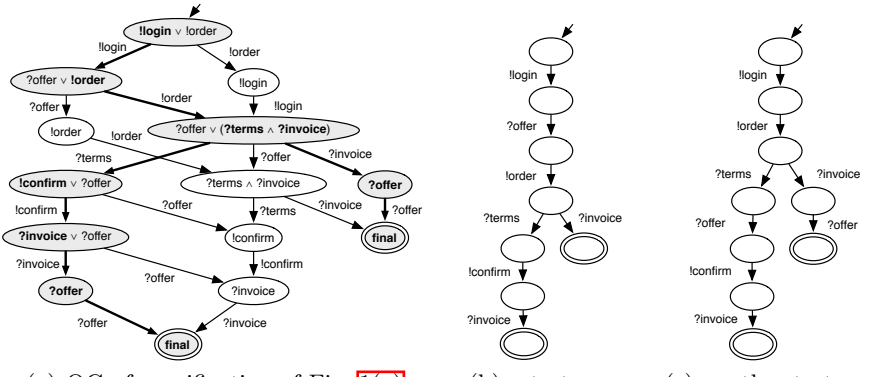
In this section, we present how test cases can be automatically generated given a service specification. Firstly, we describe how all test cases can be compactly characterized. Among the test cases, some are redundant; that is, their test result are already implied by other test cases. We conclude this section by defining some criteria when a test case can be considered redundant and show how this redundancy criterion can be used to minimize the generated test suite.

3.1 Characterizing Conformant Services

To formally reason about the behavior of a service, an exact mathematical model is needed. In this paper, we use *open nets* [9], a special class of Petri nets [10]. For BPEL, a formal semantics based on open nets was specified which allows to translate BPEL processes into open nets [11]. Furthermore, open nets can be translated back to BPEL processes [12]. As both translations are implemented[2], results in the area of open nets can be easily applied to real-life Web services.

A fundamental correctness criterion for services is *controllability* [13]. A service is controllable if there exists a partner service such that their composition (i.e., the choreography of the service and the partner) is free of deadlocks.

Controllability can be decided constructively: If a partner service exists, it can be automatically synthesized [14]. Furthermore, there exists one canonic *most permissive* partner which simulates any other partner service. The converse does, however, not hold; not every simulated service is a correct partner service itself. To this end, the most permissive partner service can be annotated with Boolean formulae expressing which states and transitions can be omitted and which parts are mandatory. This annotated most permissive partner service is called an *operating guideline* [15].



(a) OG of specification of Fig. 1(a) (b) a test case (c) another test case

Fig. 4. The operating guideline (OG) (a) describes all 1151 test cases, e. g., (b) and (c). The former can be used to show that the implementation in Fig. 3(b) does not comply to the specification of Fig. 1(a).

The operating guideline of the specification of Fig. 1(a) is depicted in Fig. 4(a). It is a finite state machine whose edges are labeled with messages sent to (preceded with "!") or received from (preceded with "?") the customer. The annotations describe for each state which outgoing edges have to be present (see [15] for further details). As mentioned earlier, subgraphs of the operating guideline are only partners if the states also fulfil the respective annotations. One example for a characterized partner is the subgraph consisting of the gray nodes and the

[2] See http://service-technology.org/tools

bold edges, also depicted in Fig. 4(b). This partner describes the behavior of the customer service of Fig. 1(c). Another test case is depicted in Fig. 4(c). In total, the operating guideline characterizes 1151 partners, i.e. 1151 rooted subgraphs fulfilling the annotations (see [15] for details).

The operating guideline of a specified service exactly characterizes the set of conformant services of this specification. Any service that is characterized by that operating guideline must also interact deadlock-free with the implementation and can therefore be seen as a test case. Using the operating guideline of the specification of a service as a characterization of the test cases postulates the requirement that the specification is controllable. If it is uncontrollable, then *any* interaction will eventually deadlock and no (positive) test cases exists. In this case, a diagnosis algorithm [16] can help to overwork the specification.

3.2 Selecting Test Cases

The operating guideline of the specification characterizes all necessary test cases to test conformance of an implementation with its specification. Unfortunately, even for small services such as the example online shop, there already exist thousands of test cases, each describing a possible customer service derived from the specification/public view. However, there are some *redundant* test cases which can be left out without reducing the quality of the test suite. Thus, even with a limited number of test cases, it is possible to detect all errors that could also be detected with the complete set of test cases.

We define redundancy as follows: A partner service T characterized by the operating guideline is a redundant test case if (1) there exists a set of partner services $T_1, \ldots, T_n$ such that each T_i is a proper subgraph of T, and (2) the union of these graphs equals T; that is, $T_1 \cup \cdots \cup T_n = T$. We can generate the test cases for the reduced test suite directly by decomposing the most permissive partner (i.e., the operating guideline without the Boolean formulae) into several smaller partner services $T_1, \ldots, T_n$, so that each partner service characterized by the operating guideline is redundant to a subset of test cases containing in the test suite. This can be realized by one depth-first search on the most permissive partner and by exploiting the operating guideline's annotations.

We illustrate the procedure on the operating guideline in Fig. 4(a). The annotation "!login ∨ !order" of the initial node demands that any test case has to start with a !login action (as the service in Fig. 4(b)), an !order action (cf. Fig. 4(c)), or both. Thereby, the service containing both actions can choose which message to send on (test) runtime. This is not desirable, because such a test case has to be executed twice: once for each action. In such a case, we can move this runtime decision to the design time of the test case and only consider those services consisting of one of the respective actions. Hence, the service with the choice between the two actions can be considered as redundant, because its behavior is covered by the other two services. Consequently, we have two kinds of test cases: ones starting with a !login action and another one starting with an !order action. By following the !login edge in the operating guideline we reach a node annotated with "?offer ∨ !order". Thus we can refine the first kind of test case

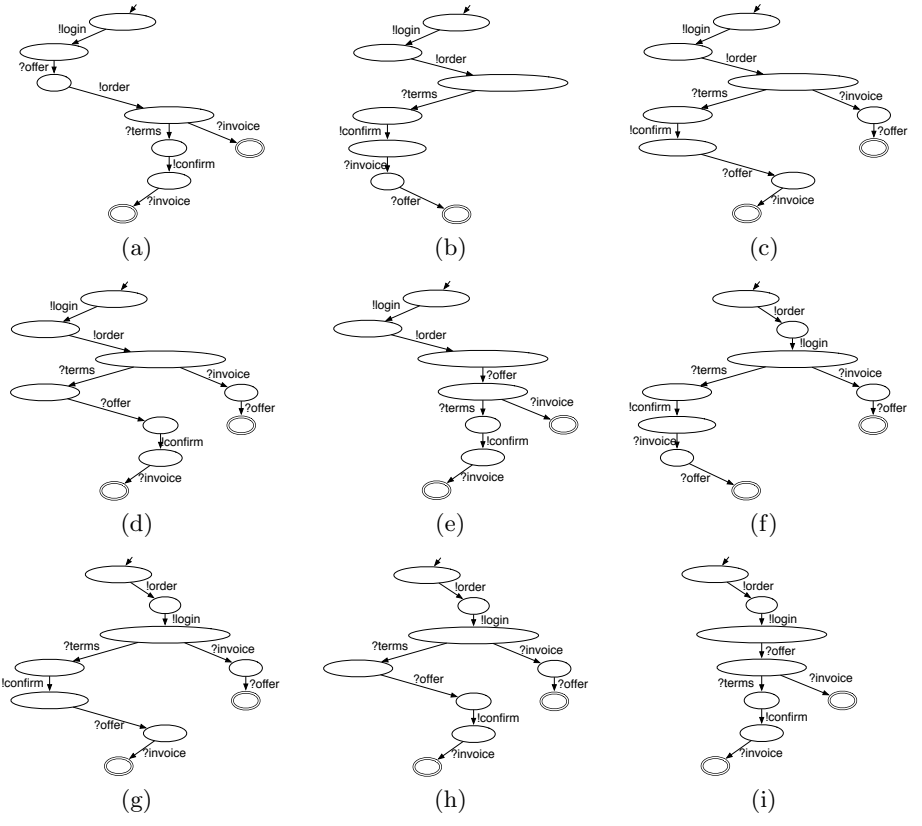


Fig. 5. Minimal test suite consisting of nine non-redundant test cases. Case (a) coincides with Fig. 4(b) and case (d) coincides with Fig. 4(c).

to those which either start with a !login followed by an ?offer or start with a !login followed by an !order. With the remaining nodes we proceed in the same manner. Finally, we will detect only nine test cases which are necessary to test the conformance of the implementation to its specification. The test cases are depicted in Fig. 5. All other test cases characterized by the operating guideline are redundant to these nine test cases. Thus the minimized test suite is complete in the sense that it is still able to detect all errors that could have been detected by all 1151 test cases.

Note our notion of redundancy will not split conjunctions. This is important as they model choices that cannot be influenced by the test case. For instance, a customer of the online shop of Fig. 1(a) cannot influence whether or not he is recognized as a premium customer and therefore must be able to receive both the terms of payment or immediately the invoice (cf. Fig. 4(b) and 4(c)).

If the specification is given as an abstract BPEL service, an existing tool chain[3] is directly applicable to translate the specification into an open net and

[3] See http://www.service-technology.org/tools

to calculate the operating guideline thereof (cf. [14]). Furthermore, the test cases can be automatically translated into BPEL processes [12]. The resulting test Web services can then be executed on a BPEL engine to test the implementation under consideration.

4 Related Work

Several works exist to systematize testing of Web services (see [17] for an overview). In [18,7], white-box test architectures for BPEL are presented. The approaches are very similar to unit testing in classical programming languages, but not applicable in our black-box setting. Furthermore, the authors give no information about how to generate test cases.

Test case generation can be tackled using a variety of approaches such as model checking [19], XML schemas [20], control or data flow graphs [21,22]. These approaches mainly focus on code coverage, but do not take the interacting nature of Web services into account. In particular, internal activity sequences are not necessarily enforceable by a test case. Therefore, it is not clear how derived test cases can be used for black-box testing in which only the interface of the service under test is available.

To the best of our knowledge, none of the existing testing approaches take stateful business protocols implemented by Web services into account, but mainly assume stateless remote procedure calls (i. e., request-response operations), see for instance [20].

Bertolino et al. [23,24] present an approach to generate test suites for Web services and also consider nonfunctional properties (QoS). Their approach, however, bases on synchronous communication and is thus not applicable in the setting of SOAs.

Finally, several tools such as the Oracle BPEL Process Manager [25] or Parasoft SOAtest [26] support testing of services, but do not specifically focus on the business protocol of a service. In contrast, Mayer et al. [7] underline the importance of respecting the business protocol during testing, but do not support the generation of test cases in their tool BPELUnit.

5 Conclusion

We presented an approach to automatically generate test cases for services. These test cases are derived from an operating guideline of a specification and can be used to test conformance of an implementation without explicit knowledge of the latter (black-box testing). We also introduced a notion of redundant test cases to drastically reduce the size of the test suite while still ensuring completeness with respect to the specified business protocol. The approach bases on open nets as formal model, and can be applied to any specification language to which a translation into open nets exists. Existing translations from BPMN [27] or UML-AD [28] into Petri nets are likely to be adjustable to open nets easily.

In this paper, we focused on business protocols in which data plays a secondary role. However, existing works such as [29,30] show that it is principally possible to add those data aspects into a formal model. As a result, we can refine not only the open net, but also the resulting operating guideline. This in turn might further reduce the number of necessary test cases as nondeterministic choices are replaced by data-driven decisions.

In future work, we also want to consider *negative test cases* to test the absence of unintended partners. Such negative test cases can be similarly derived from the operating guideline. For example, annotations can be wilfully left unsatisfied by a test case to "provoke" deadlocks during a test. Furthermore, we plan to validate our test case generation in a BPEL test architecture such as [18,7] using real-life case studies.

Beside black-box testing, the service test approach is also applicable to several other settings. Firstly, the test case generation can support *isolated testing* of services. In this case, we can use the specification to not only derive test cases, but also to synthesize stub implementation of third-party services. These stub implementations follow the business protocol as specified by the specification, but avoid possible resulting costs of third-party calls. Isolated testing might also help to locate occurring errors more easily, because we can rely on the synthesized mock implementations to conform to the business protocol.

Secondly, the completeness of the operating guideline can be used to support *substitutability* of services. When implementation I of a service is substituted by a new implementation I', we can use the same test cases to check conformance of I with the specification to check conformance of I' with that specification. In case a test fails, we can conclude that I' should not be replaced by I'.

A third scenario are inter-organizational business processes. In this setting, several parties agree on a *contract* which can be partitioned into several services. Each service then is implemented by a party. Then, conformance between the contract (a special kind of a public view) and the implementation can be tested for each party. Additionally, a party can use the global contract to derive test cases to test whether another party's implementation conforms to the contract.

Acknowledgements. Niels Lohmann is funded by the DFG project "Operating Guidelines for Services" (WO 1466/8-1).

References

1. Papazoglou, M.P.: Agent-oriented technology in support of e-business. Communications of the ACM 44(4), 71–77 (2001)
2. Martens, A.: Verteilte Geschäftsprozesse – Modellierung und Verifikation mit Hilfe von Web Services. PhD thesis, Humboldt-Universität zu Berlin, Berlin, Germany (2003) (in German)
3. Papazoglou, M.P.: Web services: Principles and technology. Prentice Hall, Englewood Cliffs (2007)
4. OMG: Business Process Modeling Notation (BPMN) Specification. Final Adopted Specification, Object Management Group (2006), http://www.bpmn.org

5. Alves, A., et al.: Web Services Business Process Execution Language Version 2.0. Committee Specification, OASIS (2007)
6. Curbera, F., Leymann, F., Storey, T., Ferguson, D., Weerawarana, S.: Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging and More. Prentice Hall PTR, Englewood Cliffs (2005)
7. Mayer, P., Lübke, D.: Towards a BPEL unit testing framework. In: TAV-WEB 2006, pp. 33–42. ACM, New York (2006)
8. König, D., Lohmann, N., Moser, S., Stahl, C., Wolf, K.: Extending the compatibility notion for abstract WS-BPEL processes. In: WWW 2008. ACM, New York (2008)
9. Massuthe, P., Reisig, W., Schmidt, K.: An operating guideline approach to the SOA. Annals of Mathematics, Computing & Teleinformatics 1(3), 35–43 (2005)
10. Reisig, W.: Petri Nets. EATCS Monographs on Theoretical Computer Science edn. Springer, Heidelberg (1985)
11. Lohmann, N.: A feature-complete petri net semantics for WS-BPEL 2.0. In: Dumas, M., Heckel, R. (eds.) WS-FM 2007. LNCS, vol. 4937, pp. 77–91. Springer, Heidelberg (2008)
12. Lohmann, N., Kleine, J.: Fully-automatic translation of open workflow net models into simple abstract BPEL processes. In: Modellierung 2008. LNI, vol. P-127. GI (2008)
13. Wolf, K.: Does my service have partners? Transactions on Petri Nets and Other Models of Concurrency (2008) (accepted for publication, preprint), http://www.informatik.uni-rostock.de/~nl/topnoc.pdf
14. Lohmann, N., Massuthe, P., Stahl, C., Weinberg, D.: Analyzing interacting BPEL processes. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) BPM 2006. LNCS, vol. 4102, pp. 17–32. Springer, Heidelberg (2006)
15. Lohmann, N., Massuthe, P., Wolf, K.: Operating guidelines for finite-state services. In: Kleijn, J., Yakovlev, A. (eds.) ICATPN 2007. LNCS, vol. 4546, pp. 321–341. Springer, Heidelberg (2007)
16. Lohmann, N.: Why does my service have no partners? In: WS-FM 2008. LNCS. Springer, Heidelberg (2008)
17. Baresi, L., Di Nitto, E. (eds.): Test and Analysis of Web Services. Springer, Heidelberg (2007)
18. Li, Z., Sun, W., Jiang, Z.B., Zhang, X.: BPEL4WS unit testing: Framework and implementation. In: ICWS 2005, pp. 103–110. IEEE, Los Alamitos (2005)
19. García-Fanjul, J., Tuya, J., de la Riva, C.: Generating test cases specifications for BPEL compositions of Web services using SPIN. In: WS-MaTe 2006, pp. 83–94 (2006)
20. Hanna, S., Munro, M.: An approach for specification-based test case generation for Web services. In: AICCSA, pp. 16–23. IEEE, Los Alamitos (2007)
21. Yan, J., Li, Z., Yuan, Y., Sun, W., Zhang, J.: BPEL4WS unit testing: Test case generation using a concurrent path analysis approach. In: ISSRE 2006, pp. 75–84. IEEE, Los Alamitos (2006)
22. Bartolini, C., Bertolino, A., Marchetti, E., Parissis, I.: Data flow-based validation of web services compositions: Perspectives and examples. In: Dehne, F., Sack, J.-R., Zeh, N. (eds.) WADS 2007. LNCS, vol. 4619, pp. 298–325. Springer, Heidelberg (2007)
23. Bertolino, A., De Angelis, G., Frantzen, L., Polini, A.: Model-based generation of testbeds for web services. In: Suzuki, K., Higashino, T., Ulrich, A., Hasegawa, T. (eds.) TestCom/FATES 2008. LNCS, vol. 5047, pp. 266–282. Springer, Heidelberg (2008)

24. Bertolino, A., De Angelis, G., Lonetti, F., Sabetta, A.: Let the puppets move! automated testbed generation for service-oriented mobile applications. In: SEAA 2008, pp. 11–19. IEEE, Los Alamitos (2008)
25. Oracle: BPEL Process Manager (2008), http://www.oracle.com/technology/products/ias/bpel
26. Parasoft: SOAtest (2008), http://www.parasoft.com
27. Dijkman, R.M., Dumas, M., Ouyang, C.: Semantics and analysis of business process models in BPMN. Information & Software Technology (2008) (accepted for publication)
28. Störrle, H.: Semantics and verification of data flow in UML 2.0 activities. Electr. Notes Theor. Comput. Sci. 127(4), 35–52 (2005)
29. Sharygina, N., Kröning, D.: Model checking with abstraction for Web services. In: Test and Analysis of Web Services, pp. 121–145. Springer, Heidelberg (2007)
30. Moser, S., Martens, A., Görlach, K., Amme, W., Godlinski, A.: Advanced verification of distributed WS-BPEL business processes incorporating CSSA-based data flow analysis. In: SCC 2007, pp. 98–105. IEEE, Los Alamitos (2007)

Detecting Behavioural Incompatibilities between Pairs of Services*

Ali Aït-Bachir[1,**], Marlon Dumas[2,***], and Marie-Christine Fauvet[1]

[1] LIG, University of Grenoble, France
{Ali.Ait-Bachir,Marie-Christine.Fauvet}@imag.fr
[2] University of Tartu, Estonia
Marlon.Dumas@ut.ee

Abstract. We present a technique to analyse successive versions of a service interface in order to detect changes that cause clients using an earlier version not to interact properly with a later version. We focus on behavioural incompatibilities and adopt the notion of simulation as a basis for determining if a new version of a service is behaviourally compatible with a previous one. Unlike prior work, our technique does not simply check if the new version of the service simulates the previous one. Instead, in the case of incompatible versions, the technique provides detailed diagnostics, including a list of incompatibilities and specific states in which these incompatibilities occur. The technique has been implemented in a tool that visually pinpoints a set of changes that cause one behavioural interface not to simulate another one.

1 Introduction

Throughout its lifecycle, the interface of a software service is likely to undergo changes. Some of these changes do not cause existing clients or peers to stop interacting properly with the service. Other changes give rise to incompatibilities. This paper is concerned with the identification of these latter changes.

Service interfaces can be seen from at least three perspectives: structural, behavioural and non-functional. The structural interface of a service describes the schemas of the messages that the service produces or consumes and the operations underpinning these message exchanges. In the case of Web services, the structural interface of a service can be described for example in WSDL [14]. The behavioural interface describes the order in which the service produces or consumes messages. This can be described for example using BPEL [14] business protocols, or more simply using state machines as discussed in this paper. The work presented here focuses on behavioural interfaces and is complementary to other work dealing with structural interface incompatibility [12].

In this paper, we present a technique for comparing two service interfaces in order to detect a series of changes that cause them not to be compatible from

* Work partly funded by the Web Intelligence Project, Rhône-Alpes French Region.
** The author was supported by a visiting PhD scholarship at University of Tartu.
*** The author is also affiliated with Queensland University of Technology, Australia.

G. Feuerlicht and W. Lamersdorf (Eds.): ICSOC 2008, LNCS 5472, pp. 79–90, 2009.

a behavioural viewpoint. We consider three types of differences between two services S1 and S2: an operation that produces a message is enabled in a state of service S2 but not in the equivalent state in S1; an operation that consumes a message is enabled in a state in S1 but not in the equivalent state in S2; and and operation enabled in a state of S1 is replaced by another operation in the equivalent state in S2. It may be that S2 allows operation Op to be invoked, but at a different point in time than S1 does. So the diagnosis our technique provides goes beyond what can be provided based purely on structural interfaces.

The paper is structured as follows. Section 2 frames the problem and introduces an example. Section 3 defines a notion of behavioural interface while Section 4 presents the incompatibility detection algorithm. Section 5 compares the proposal with related ones and Section 6 concludes and sketches future work.

2 Motivation

As a running example, we consider a service that handles purchase orders processed either online or offline. Figure 1 depicts three behavioural interfaces related to this example. These behavioural interfaces are described using UML activity diagrams notations that capture control-flow dependencies between message exchanges (i.e. activities for sending or receiving messages). The figure distinguishes between the *provided* interface that a service exposes, and its *required* interface as it is expected by its clients or peers. Specifically, the figure shows the provided interface P of an existing service (see left-hand side of the figure). This service interacts with a client application that requires an interface R (shown in the centre of the figure). We consider the scenario where another service which satisfies similar needs, but whose interface is P' (shown in the right-side of the figure). In this setting, the questions that we address are: (i) does the differences between P and P' cause incompatibilities between P' and P's existing client(s); and (ii) if so, which specific changes lead to these incompatibilities. As mentioned above, we consider three types of changes: addition and deletion of an operation enabled in a given state of P, and replacement of an operation enabled in a state of P with a different operation enabled in a corresponding state in P'.[1]

In Figure 1, we observe that the flow that loops from *Receive OfflineOrder* back to itself in P does not appear in P'. In other words, according to P''s interface, customers are not allowed to alter offline orders. This is a source of incompatibility since clients that rely on interface P may attempt to send messages to alter their offline order but the service (with interface P') does not expect a new order after the first order. On the other hand, message *Shipment Tracking Number* (STN) has been replaced in P' by message *Advance Shipment Notice* (ASN). This difference will certainly cause an incompatibility vis-a-vis of existing client applications and peer services. Note also that the possibility of paying by bank transfer has been added to the branch of the behavioural interface that deals with online orders. However, this addition does not lead

[1] We use the terms operation and message interchangeably, while noting that strictly speaking, messages are events that initiate or result from operations.

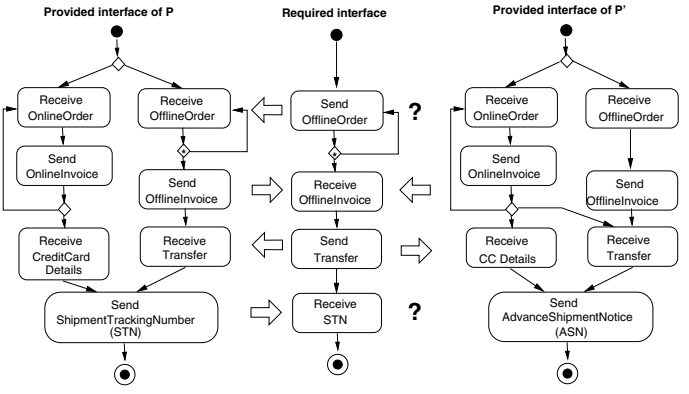


Fig. 1. P and P' provided interfaces

to an incompatibility since existing client applications or peer services are not designed to use this option. This later case shows that an incompatibility only arises when P' offers less options than P. In technical terms, a change between P' and P only leads to an incompatibility if it causes P' not to simulate P.

3 Modeling Service Behaviour

The proposed technique for detecting incompatibilities is based on the comparison of behavioural interfaces capturing order dependencies between messages sent and received by a service. We only consider message names, without inspecting the schema of these messages.

Following [3,10], we adopt a simple yet effective approach to model service behaviour using *Finite State Machines* (FSMs). Techniques exist to transform behavioural service interfaces defined in other languages (e.g. BPEL) into FSMs (see for example the WS-Engineer tool [7]), and therefore the choice of FSM should not be seen as a restriction. What we can note is that during the transformation from behaviour description languages that support parallelism (e.g. BPEL) into FSMs, parallelism is encoded in the form of interleaving of actions. For example, the parallel execution of activities a and b is encoded as a choice between 'a followed by b' and 'b followed by a'. In the FSMs we consider, transitions are labelled with message exchanges. When a message is sent or received, the corresponding transition is fired. Figure 2 depicts FSMs of provided interfaces P and P' of the running example presented in Section 2. The message m is denoted by $>m$ (resp. $<m$) when it is sent (resp. received). Each conversation initiated by a client starts an execution of the corresponding FSM.

Definitions and notations: An FSM is a tuple (S, L, T, s_0, F) where: S is a finite set of states, L a set of events (actions), T the transition function ($T : S \times L \longrightarrow S$). s_0 is the initial state such that $s_0 \in S$, and F the set of final states such that $F \subset S$. The transition function T associates a source state $s_1 \in S$ and

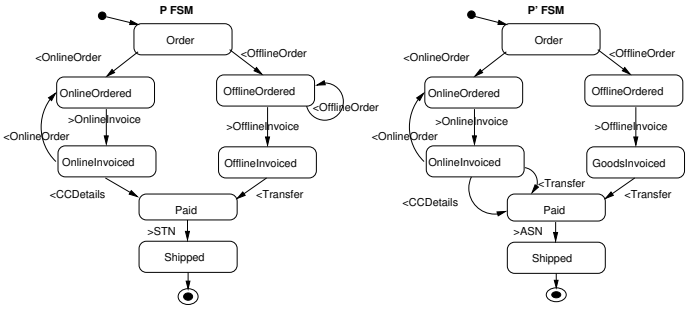


Fig. 2. FSMs of two provided interfaces

an event $l_1 \in L$ to a target state $s_2 \in S$. In this model, a transition is defined as a tuple containing a source state, a label and a target state.

We assume synchronous communication. While in reality Web service communication is not always synchronous, synchronous communication provides, to a certain extent, a suitable basis for analysing service behaviour. First of all, synchronous communication is more restrictive than asynchronous communication. Therefore, incompatibilities that arise within the asynchronous case arise in the synchronous case as well. Second, for a relatively large class of interfaces, it has been shown that adopting the synchronous communication model leads to the same analysis results than adopting the asynchronous model [6].

Another assumption is that we focus on interfaces that expose only externally visible behaviour. In particular, internal actions or timeouts do not appear in the service interface unless they are externalized as messages.

Below, we use the following notations (examples refer to the FSM P depicted in the left side in Figure 2):

- $s\bullet$ is the set of outgoing transitions from s.
  (e.g. OnlineInvoiced$\bullet$ = {(OnlineInvoiced,<CCDetails,Paid),(OnlineInvoiced, <OnlineOrder,OnlineOrdered) }).
- $t\circ$ is the target state of the transition t.
  (e.g. (OnlineInvoiced,<CCDetails,Paid)$\circ$ =Paid).
- Label(t) is the label of the transition t.
  (e.g. Label((OnlineInvoiced,<CCDetails,Paid))=<CCDetails)
- $\| X \|$: set cardinality of a set X.
- The $\circ$ operator (respectively $\bullet$) is generalised to a set of transitions (respectively states). For example, if $T = \bigcup_{i=1}^{n}\{t_i\}$ then $T\circ = \bigcup_{i=1}^{n}\{t_i\circ\}$; where $n = \| T \|$. Similarly, operator *Label* is generalized to a set of transitions.

4 Detection of Changes

To detect changes, P and P' are traversed synchronously starting from their respective initial states s_0 and s'_0. The traversal seeks for two states s and s' (belonging respectively to P and P') such that the sub-automaton starting from

s in P and the one starting from s' in P' are *incompatible*. The traversal algorithm is described in section 4.4. But first, we discuss the conditions that need to be evaluated to diagnose each type of change: deletion (see Section 4.1), addition (see Section 4.2) and modification of an operation (see Section 4.3).

4.1 Deletion of an Operation

Figure 3 illustrates a situation where a deletion can be diagnosed. It shows two FSMs: one corresponding to a service (P) and the other to another service P'. We observe that all operations enabled in state $S1'$ are also enabled in state S. On the other hand, there is an operation (namely $>R(m)$) enabled in state S that has no match in state $S1'$. So we can conclude that operation $>R(m)$ has been deleted from this particular state.

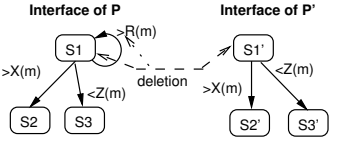


Fig. 3. First case where a deletion is diagnosed

Figure 4 depicts a second scenario where a deletion can be diagnosed. We first note that the above condition does not hold: not all operations enabled in $S1'$ are enabled in $S1$. Indeed, operation $<Z(m)$ is enabled in $S1'$ but not in $S1$. At the same time, operation $>X(m)$ is enabled in $S1$ but it is not enabled in $S1'$. There are two possibilities for this mismatch: either operation $>X(m)$ has been modified and has become $<Z(m)$, or operation $>X(m)$ has been deleted altogether. In this example, we can discard the former possibility because $<Z(m)$ appears downstream in the interface FSM of P (it is the label of the outgoing transition of state $S2$). Thus, $<Z(m)$ can not be considered to be a replacement for $>X(m)$. So we conclude that $>X(m)$ has been deleted.

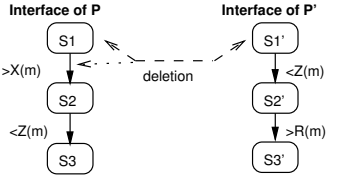


Fig. 4. Second case where a deletion is diagnosed

Once this deletion is detected, the state pair to be examined next in the comparison of P and P' is $(S2, S1')$. In other words, when deleting a transition, we jump to its target state and continue looking for other changes that may be sources of incompatibilities. For reporting purposes, the deletion is denoted by a tuple $(S1, >X(m), S1', null)$, meaning that operation $>X(m)$ enabled in $S1$ is replaced by the 'null' value in $S1'$. Formally, when comparing two interface

FSMs P and P', a deletion is diagnosed in a pair of states s and s' (respectively belonging to P and P') if the following condition holds (each part of this condition is explained below):

$$\|Label(s\bullet) - Label(s'\bullet)\| \geqslant 1 \wedge \|Label(s'\bullet) - Label(s\bullet)\| = 0 \quad (1)$$
$$\vee\ \exists t \in s\bullet, \exists t' \in s'\bullet : Label(t) \notin Label(s'\bullet) \wedge ExtIn(t', (t\circ)\bullet) \quad (2)$$

A deletion is detected in state pair (s, s') in two cases. The first one (line 1) is when every outgoing transition of s' can be matched to an outgoing transition of s, but on the other hand, there is an outgoing transition of s that can not be matched to a transition of s'. A second case is when there exists a pair of outgoing transitions t and t' (of states s and s' respectively) such that: (i) transition t can not be matched to any outgoing transition of s'; and (ii) the label of t' occurs somewhere in the FSM rooted at the target state of t (line 2).[2] This second condition is tested in order to determine whether the non-occurrence of t's label among the outgoing transitions of s' should indeed be interpreted as a deletion, as opposed to a modification or an addition. To check if a transition label occurs somewhere in the FSM rooted at the target of a given transition, we use the following recursive boolean function: $ExtIn(t, T) \equiv T \neq \emptyset \wedge (Label(t) \in Label(T) \vee \bigcup_{i=1}^{\|T\|} ExtIn(t, (T_i\circ)\bullet))$. In other words, $ExtIn(t, T)$ (where t is a transition and T is a set of transitions) evaluates to true if either transition t's label appears among the labels of transitions in T ($Label(t) \in Label(T)$) or, there exists a transition taken in T which has a target state whose set of outgoing transitions (namely $T1$) is such that $ExtIn(t, T1)$ evaluates to true. The way it is defined, this recursive function does not converge if the FSM has cycles, but it can be trivially extended to converge by adding an input parameter to store the set of visited states and to ensure that each state is only visited once.

4.2 Addition of an Operation

We now consider the diagnosis of an incompatibility resulting from the addition of an operation in state pair $(S1, S1')$. The simplest case is when all the operations enabled in state $S1$ are also enabled in $S1'$ but not the opposite. This is the case for example of *<Z(m)* in Figure 5. This particular addition however does not lead to an incompatibility because what it does is that it allows a service implementing P' to accept an additional message that a service implementing P would not accept. An existing client of P would simply not send this message. Thus, existing clients of P can interact with P', even though such clients would never use the branch starting with transition labelled $<Z(m)$. On the other hand, if we replaced $<Z(m)$ with $>Z(m)$, the addition would give rise to an incompatibility, because the service implementing P' may try to produce a message *Z(m)* that a client of P would not accept.

[2] By *FSM P rooted at s* we mean FSM P in which the initial state is set to be s. This means that we ignore any state or transition that is not reachable from s.

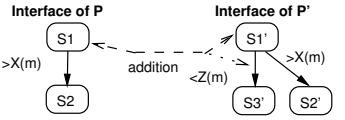


Fig. 5. First case where an addition is diagnosed

Figure 6 illustrates another case where an addition can be diagnosed. Operation $>X(m)$ is enabled in state $S1$ and is not enabled in $S1'$. On the other hand, operation $<Y(m)$ is enabled in state $S1'$ but not in $S1$ and operation $>X(m)$ is enabled in a state downstream along the transition labelled $<Y(m)$. Thus we can conclude that operation $<Y(m)$ has been added. This addition constitutes an incompatibility regardless of whether $Y(m)$ is sent or received, because the non-occurrence of $Y(m)$ would prevent the execution of a service implementing P' to progress along the branch leading to the state where $>X(m)$ can occur.

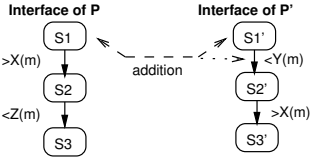


Fig. 6. Second case where an addition is diagnosed

When an addition is detected in a state pair (S, S'), the synchronous traversal of the two FSMs advances along the added transition. In the case of Figure 5 this means that $(S1, S2')$ should be visited next, while in the case of Figure 6, state pair $(S1, S3')$ should be visited next – in addition to $(S2, S2')$ since this latter state pair can be reached by taking transitions $>X(m)$ synchronously. For reporting purposes, the addition of an operation $<Y(m)$ is denoted by a tuple $(S1, null, S1', <Y(m))$. Formally, an addition of an operation is diagnosed in state pair (s,s') if the following condition holds:

$$(\|Label(s\bullet) - Label(s'\bullet)\| = 0 \wedge \|Label(s'\bullet) - Label(s\bullet)\| \geqslant 1) \quad (3)$$

$$\vee\ \exists t \in s\bullet, \exists t' \in s'\bullet : Label(t') \notin Label(s\bullet) \wedge ExtIn(t, (t'\circ)\bullet) \quad (4)$$

An addition is detected in two cases. The first case (line 3) is when there exists an outgoing transition of s' whose label does not match any of the labels of the outgoing transitions of s, while at the same time, every outgoing transition of s can be matched to an outgoing transition of s'. In this case, we need to additionally check whether the added operation corresponds to a "send" or a "receive", since an added "receive" does not constitute an incompatibility in this case. The second case (line 4) is when there exists a pair of outgoing transitions t and t' (of states s and s' respectively) with different labels and such that the label of t appears in the FSM rooted at the target of transition t'.

4.3 Modification of an Operation

Figure 7 shows a situation where we can diagnose that operation *>X(m)* has been replaced by operation *>Y(m)* (i.e. a modification). We can make this diagnosis because operation *>X(m)* is enabled in $S1$ but not in $S1'$, and conversely *>Y(m)* is enabled in $S1'$ but not in $S1$. Moreover, the transition labelled *>X(m)* can not be matched to a transition t' in state $S1'$ such that operation *>X(m)* occurs downstream along the branch starting with t', and symmetrically, *>Y(m)* can not be matched with a transition t of state $S1$ such that *>Y(m)* occurs downstream along the branch starting with t. Thus we can not diagnose that *>X(m)* has been deleted, nor can we diagnose that *>Y(m)* has been added.

In this case, the pairing of transition *>X(m)* with transition *>Y(m)* is arbitrary. If state $S1'$ had a second outgoing transition labelled *>Z(m)*, we would equally well diagnose that *>X(m)* has been replaced by *>Z(m)*. Thus, when we diagnose that *>X(m)* has been replaced by *>Y(m)*, all that we capture is that *>X(m)* has been replaced by another operation, possibly *>Y(m)*. The output produced by the proposed technique should be interpreted in light of this.

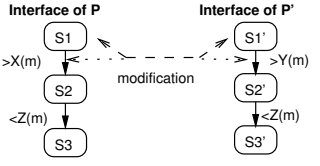


Fig. 7. Diagnosis of a modification/replacement

For reporting purposes, a modification (replacement) of $>X(m)$ into $>Y(m)$) is denoted by tuple $(S1, >X(m), S1', >Y(m))$. The state pair to be visited next in the synchronous traversal of P and P' is such that both transitions involved in the modification are traversed simultaneously. In this example, $(S2, S2')$ should be visited next. Formally, a modification is diagnosed in state pair (s,s') if the following condition holds:

$$\exists t1 \in s\bullet, \exists t1' \in s'\bullet : Label(t1) \notin Label(s'\bullet) \wedge Label(t1') \notin Label(s\bullet) \quad (5)$$

$$\wedge\neg\exists t2 \in s\bullet : ExtIn(t1', (t2\circ)\bullet)) \wedge \neg\exists t2' \in s'\bullet : ExtIn(t1, (t2'\circ)\bullet)) \quad (6)$$

4.4 Detection Algorithm

The algorithm implementing the detection of changes is detailed in Figure 8. Given two interface FSMs P and P', the algorithm traverses P and P' synchronously starting from their respective initial states s_0 and s'_0. At each step, the algorithm visits a state pair consisting of one state from each of the two FSMs. Given a state pair, the algorithm determines if an incompatibility exists and if so, it classifies it as an addition, deletion or modification. If an *addition* is detected the algorithm progresses along the transition of the added operation in P' only. Conversely, if the change is a *deletion*, the algorithm progresses along the transition of the deleted operation in P only. However, if a *modification*

```
1  Detection (Pi: FSM; Pj: FSM ): {Change}
2  { Detection(Pi,Pj) returns a set of tuples of changes represented as tuple of the form < si, ti, sj, tj >
   where si and sj are states of Pi and Pj respectively, while ti and tj are either null values or outgoing
   transitions of si and sj respectively }
3  setRes: {Change}; { result variable }
4  si,sj: State ; { auxiliary variables }
5  visited, toBeVisited : Stack of statePair; { pairs of states that have been visited / must be visited }
6  si ← initState(Pi) ; sj ← initState(Pj)
7  toBeVisited.push((si,sj))
8  while notEmpty(toBeVisited)
9      (si, sj) ← toBeVisited.pop();
10     visited.push( (si, sj) ) { add the current state pair to the visited stack }
11     combEqual ← {(ti, tj) ∈ si• × sj• | Label(ti) = Label(tj)} { pairs of matching transitions }
12     difPiPj ← {ti ∈ si• | Label(ti) ∉ Label(sj•)}; difPjPi ← {tj ∈ sj• | Label(tj) ∉ Label(si•)}
13     combPiPj ← difPiPj × difPjPi; { all pairs of outgoing transitions of si and sj that do not have a
                                       match }
14     If ||difPiPj|| ⩾ 1 and ||difPjPi|| = 0 then                                          { deletion }
15         For each t in difPiPj do setRes.add(< si, t, sj, null>)
16             If((t○, sj) ∉ visited) then toBeVisited.push((t○, sj))
17     If ||difPjPi|| ⩾ 1 and ||difPiPj|| = 0 then                                          { addition }
18         For each t in difPjPi do
19             If (polarity(t) = 'send') then setRes.add(< si, null, sj, t>) { otherwise this addition does not
                                                                               lead to incompatibility }
20             If ((si, t○) ∉ visited) then toBeVisited.push((si, t○))
21     For each (ti, tj) in combPiPj do
22         If ExtIn(ti, (tj○)•) then                                                        { addition }
23             setRes.add(< si, null, sj, tj>)
24             If ((si, tj○) ∉ visited) then toBeVisited.push((si, tj○))
25         If ExtIn(tj, (ti○)• ) then                                                       { deletion }
26             setRes.add(< si, ti, sj, null,'deletion'>)
27             If ((ti○, sj) ∉ visited) then toBeVisited.push((ti○, sj))
28         If ( (¬∃tj' ∈ sj• : ExtIn(ti, (tj'○)•)) ∧ (¬∃ti' ∈ si• : ExtIn(tj, (ti'○)•))) then   { modif. }
29             setRes.add(< si, ti, sj, tj>)
30             if((ti○, tj○) ∉ visited) then toBeVisited.push((ti○, tj○))
31     For each (ti, tj) in combEqual do If ((ti○, tj○) ∉ visited) then toBeVisited.push((ti○, tj○))
32 Return setRes
```

Fig. 8. Detection algorithm

is detected, the algorithm progresses along both FSMs simultaneously. While traversing the two input FSMs, the algorithm accumulates a set of changes represented as tuples of the form (s, t, s', t'), as explained previously.

The algorithm proceeds as a depth-first algorithm over state pairs of the compared FSMs. Two stacks are maintained: one with the visited state pairs and another with state pairs to be visited (line 5). These state pairs are such that the first state belongs to the FSM of Pi while the second state belongs to the one of Pj. The first state pair to be visited is the one containing the initial states of Pi and Pj (line 6). Once a pair of states is visited it will not be visited again. To ensure this, the algorithm uses the variable *visited* to memorize the already visited state pairs (line 10).

Labels that appear both in the outgoing transitions of si and in the outgoing transitions of sj are considered as unchanged. Thus, a set of state pairs is built where states are target states of common labels (line 11). Also, the algorithm reports all differences between the outgoing transitions of si and the outgoing transitions of sj (line 12). The two set differences of transitions are put in two variables *difPiPj* (transitions whose labels belong to $Label(si\bullet)$ but do not belong to $Label(sj\bullet)$) an *difPjPi* (transitions whose labels belong to $Label(sj\bullet)$ but do not belong to $Label(si\bullet)$). Line 13 calculates all combinations of transitions whose labels are not in common among $Label(si\bullet)$ and $Label(sj\bullet)$.

Lines 14 to 16 detect a deletion when an outgoing transition of si does not match any transition in $sj\bullet$. The result is a set of tuples $< si, t, sj, null >$ where t is one of the outgoing transitions of si whose label does not appear in any of sj's outgoing transitions. The detection of an addition is quite similar to the detection of a deletion (lines 17 to 20).

Variable *combPiPj* contains transition pairs such that the label of the first transition ti belongs to $si\bullet$ but does not belong to $Label(sj\bullet)$ while the label of the second transition tj belongs to $sj\bullet$ but not to $Label(si\bullet)$. For each such transition pair, the algorithm checks the conditions for diagnosing an *addition* (lines 22 to 24), a *deletion* (lines 25 to 27) or a *modification* (lines 28 to 30). Finally, the algorithm progresses along pairs of matching transitions, i.e. pairs of transitions with identical labels (line 31). The algorithm has a worst-case complexity quadratic on the total number of transitions in both FSMs.

The detection algorithm is implemented in a tool called *BESERIAL* [1] available at `http://www-clips.imag.fr/mrim/User/ali.ait-bachir/webServices/webServices.html`. Figure 9 shows the output of the compatibility analysis performed by BESERIAL on the example introduced in Section 2. Here, *Process2* is the more recent version of the interface. The operation that allows clients to update an offline order has been deleted ($<OfflineOrder$). We can see a state pair $(offlineOrdered, \_offlineOrdered)$ linked by a dashed edge labelled with the change *deletion*. The deleted operation is $<OfflineOrder$ shown by a dotted arrow. Other changes (*addition* of $<Transfer$ and modification of $>STN$ by $>ASN$) are pinpointed as well.

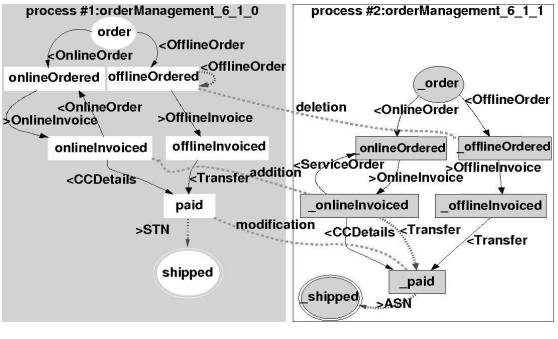


Fig. 9. Graphical output of BESERIAL on the running example

5 Related Work

Compatibility test of interfaces has been widely studied in the context of Web service composition. Most approaches that deal with the behavioural dimension of interfaces rely on equivalence and similarity techniques to check, *at design time*, whether or not interfaces described for instance by automata are compatible [4,2]. These techniques usually rely either on trace equivalence checking or on (bi-)simulation algorithms [9]. However, these approaches do not deal with

pinpointing exact locations of incompatibilities. In [13], the authors address the issue of runtime replaceability of services by extending the notions of design-time replaceability defined in [2] which are based on trace comparison. Again, this work does not aim at pinpointing a complete set of differences between service behaviours as we do in our work.

Recent research has addressed interface similarity measures issues. In [8], the author presents a similarity measure for labeled directed graphs inspired by the simulation and bi-simulation relations on labeled transition systems. The author applies this technique to detect and correct deadlocks. Other algorithms based on graph-edit distances have been applied to service discovery in [5], but do not pinpoint behavioural differences between services as our work does.

In [11], the authors propose an operator *match* which is a similarity function comparing two interfaces by finding correspondences between models. The similarity measure is a heuristics which returns a value calculated according to changes involving the addition or the deletion of an operation. However, the result does not pinpoint the exact location of these changes. In [15], the authors propose an approach to business process matchmaking based on automata extended with logical expressions associated to states. Their algorithm determines if the languages of two automata (which model two business processes) have a non-empty intersection. This technique for detecting process differences returns a boolean output. It does not provide detailed diagnostics such as pinpointing specific states of the two services are different, which is the goal of our work.

6 Conclusion and Future Work

We presented a technique to detect changes (*addition*, *deletion* or *modification* of an operation) that give rise to behavioural incompatibilities between two services. The originality of this technique is that the detection algorithm does not stop at the first incompatibility encountered but tries to seek further to identify a series of incompatibilities between two services.

Ongoing work aims at extending BESERIAL towards two directions: (i) detecting complex types of incompatibilities (e.g. the order of two operations is swapped or an entire branch is deleted); and (ii) assisting service designers in determining how to address an incompatibility. Also, BESERIAL currently assumes synchronous communication. Future work will aim at supporting asynchronous communication. We foresee that the incompatibility detection algorithm can be extended in this direction by maintaining a buffer of unconsumed messages during the traversal, along the lines of [10].

References

1. Aït-Bachir, A., Dumas, M., Fauvet, M.-C.: BESERIAL: Behavioural Service Interface Analyser. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, Springer, Heidelberg (2008)

2. Benatallah, B., Casati, F., Toumani, F.: Representing, analysing and managing web service protocols. Data and Knowledge Engineering 58(3), 327–357 (2006)
3. Beyer, D., Chakrabarti, A., Henzinger, T.A.: Web service interfaces. In: Proc. of the 14th WWW int. conf., Japan. ACM, New York (2005)
4. Bordeaux, L., Salaün, G., Berardi, D., Mecella, M.: When are two web services compatible? In: Shan, M.-C., Dayal, U., Hsu, M. (eds.) TES 2004. LNCS, vol. 3324, pp. 15–28. Springer, Heidelberg (2005)
5. Corrales, J.C., Grigori, D., Bouzeghoub, M.: BPEL processes matchmaking for service discovery. In: Meersman, R., Tari, Z. (eds.) OTM 2006. LNCS, vol. 4275. Springer, Heidelberg (2006)
6. Fu, X., Bultan, T., Su, J.: Synchronizability of conversations among web services. IEEE Transactions on Software Engineering 31(12) (2005)
7. Foster, H., Uchitel, S., Magee, J., Kramer, J.: Ws-engineer: A tool for model-based verification of web service compositions and choreography. In: Proc. of the IEEE Int. Conf. on Software Engineering, China (2006)
8. Lohmann, N.: Correcting deadlocking service choreographies using a simulation-based graph edit distance. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240. Springer, Heidelberg (2008)
9. Martens, A., Moser, S., Gerhardt, A., Funk, K.: Analyzing compatibility of bpel processes. In: Proc. of the Advanced Int. Conf. on Telecom. and Int. Conf. on Internet and Web Applications and Services, French Caribbean. IEEE, Los Alamitos (2006)
10. Motahari-Nezhad, H.R., Benatallah, B., Martens, A., Curbera, F., Casati, F.: Semi-automated adaptation of service interactions. In: Proc. of the 16th WWW Int. Conf., Canada. ACM, New York (2007)
11. Nejati, S., Sabetzadeh, M., Chechik, M., Easterbrook, S., Zave, P.: Matching and merging of statecharts specifications. In: Proc. of the 29th Int. Conf. on Software Engineering, USA. IEEE Computer Society, Los Alamitos (2007)
12. Ponnekanti, S.R., Fox, A.: Interoperability among independently evolving web services. In: Jacobsen, H.-A. (ed.) Middleware 2004. LNCS, vol. 3231, pp. 331–351. Springer, Heidelberg (2004)
13. Ryu, S.H., Casati, F., Skogsrud, H., Benatallah, B., Saint-Paul, R.: Supporting the dynamic evolution of web service protocols in service-oriented architectures. ACM Transactions on the Web 2(2), 46 (2008)
14. Weerawarana, S., Curbera, F., Leymann, F., Storey, T., Ferguson, D.: Web Services Platform Architecture. Prentice-Hall, Englewood Cliffs (2005)
15. Wombacher, A., Fankhauser, P., Mahleko, B., Neuhold, E.: Matchmaking for business processes based on choreographies. In: Proc. of the Int. Conf. on Multimedia and Expo., Taipei, Taiwan. IEEE Computer Society Press, Los Alamitos (2004)

On Supporting the Design of Human-Provided Services in SOA*

Daniel Schall, Christoph Dorn, Hong-Linh Truong, and Schahram Dustdar

Distributed Systems Group, Vienna University of Technology
Argentinierstr 8/184-1, 1040 Vienna, Austria
{schall,dorn,truong,dustdar}@infosys.tuwien.ac.at

Abstract. Collaboration platforms evolve into service-oriented systems, promoting composite and user-enriched services. The problem we address in this paper is the support of human interactions in SOA. Current collaboration tools do not support humans to specify different interaction interfaces (services), which can be reused in various collaborations. We focus on the design of Human-provided Services (HPS). Our contributions center around two main aspects of human interactions in SOA: (i) an approach for designing service interfaces embodying human activities as actions offered by Web (HPS) users; (ii) a tagging model for activities and services to recommend resources in the design process. We discuss techniques for mapping human activities onto Web services. We present a recommendation algorithm that is based on collaborative tagging of resources in SOA. Our algorithm helps to determine suitable resources drawn from properties of user preferences and measured similarity of human activities and actions.

1 Introduction

The global nature of the Web promotes access to the knowledge of an immense large number of people. The Web enables the participation of users in collaborations by various means. Currently, users interact with the Web and share content, their knowledge, and opinions, etc. through Web sites, forums, Wikis, or blogs. We have taken user participation on the Web a step further by introducing Human-provided Services (HPS) utilizing SOA technologies and Web services [1]. People can publish activities and their capabilities as Web services, creating a Web of HPSs interwoven with current Wikis, blogs, social networks and enterprise services. We believe that HPS will be the future trend of human participation on the Web and enterprise collaboration. Therefore, supporting users in designing HPSs in an easy manner is an important issue. HPSs are exposed as real Web service interfaces acting as interaction interfaces toward humans. From the user's point of view, services are represented as *activities* and *actions* a user can perform in SOA-based collaboration environments. We believe that users should be empowered to define and provide services. HPS use cases include

* Part of this work was supported by the EU project inContext (IST-034718).

G. Feuerlicht and W. Lamersdorf (Eds.): ICSOC 2008, LNCS 5472, pp. 91–102, 2009.

i) obtaining input or opinion from experts, ii) interactions with infield workers to perform certain tasks, iii) mass collaboration of globally distributed teams or communities by gathering results/output from users depending on expertise, geographic location, and availability.

We have designed and implemented a framework [2] with the aim of supporting HPSs as *user-contributions* in collaborations. While approaches such as BPEL4People [3] target the support of human interactions as part of business processes (i.e., workflows) by designing and executing a set of *human tasks* (cf. WS-HumanTask [4]), HPS enables the definition of services offered by humans independent of any process or task. We distinguish between **(a)** service — at the technical level encompassing the definition of domain specific interaction interfaces and input/output parameters — defined by users and/or communities and **(b)** tasks to model the demanded (human) input as part of a workflow.

The goal of HPS is to provide a framework and tools enabling the engagement of humans in distributed, large-scale collaborations using SOA. To enable human participation through HPS, users must be able to utilize tools to design and model their participations. These tools must be simple yet powerful enough to deal with complexities in the service-design process. To date, most effort has been spent on tools for Web service professionals and developers which however cannot be used by novice users. Mashup editors[1] are examples of how simple tools can facilitate user participations and user driven processes by gathering and aggregating different sources of knowledge. We argue that similar tools should be provided for HPS design, enabling novice users to design their personal services, thus creating a novel blend of SOA comprising human and software services. In our previous work, we introduced a middleware enabling HPSs (i.e., registry, discovery, and interactions). In this paper, we present methods and tools supporting the user in the design of HPSs in SOA-based environments. We analyze the complexity and challenges of the design process and present our solution.

Our goal is to provide powerful yet simple tools for users to define and provide services. Such tools should automatically generate all the artifacts needed to allow users to fully participate in interactions in SOA using Web services. This paper presents an architecture and its implementation allowing humans to design services for ad-hoc and process-centric collaborations, with the following key contributions, (i) a methodology for incorporating human interactions in SOA, (ii) design and implementation of the HPS designer architecture, giving users the ability to design activity-centric interfaces which can be translated into low level services descriptions (e.g., described in WSDL), and finally (iii) a method for helping users in providing the right service by using tagging methods.

This paper is organized as follows: in Sec. 2 we outline our approach and propose our solution to designing HPSs in SOA. We given an overview of the HPS framework in Sec. 3, followed by Sec. 4 describing transformations and mappings of human activities onto services. We introduce our recommendation algorithm in Sec. 5 and show implemented tools for ranking and design in Sec. 6. Finally, we discuss related work in Sec. 7 and conclude the paper in Sec. 8.

[1] E.g., http://pipes.yahoo.com/pipes/

2 Collaboratively Designing HPS

Dynamic collaborations typically take place using various communication channels and tools. The HPS middleware is a platform targeting SOA-based collaboration scenarios involving both human and software services. In this section, we first discuss the challenges in designing HPSs and present our approach. Then, we provide an overview of the steps in the design process and show how users are supported in finding/reusing exsting service artifacts.

- *Interface transformation and generation:* Designing and providing a service should be as simple as writing a "blog entry". Mapping human activities onto Web services is challenging. Novice users have to be supported in designing services in an easy and intuitive manner by hiding underlying complex processes, constituting automatic service interface generation and translation of service interfaces (e.g., WSDL) into GUI representations. Since standard Web service technologies are used — at the technical level — to enable HPS, versatile collaborations can be supported including interaction between humans as well as using HPSs in, for example, formalized processes. Human input in a process is depicted in Fig. 1 (a) as `Sends request`.
- *Recommendations for HPS Design:* We argue that humans should be able to design and provide their capabilities as services. Many HPSs may be available and registered as services, potentially with different interface characteristics and expertise of users. Users have to be supported in the design process by recommending resources and interfaces which may already exist. We propose tagging mechanism helping users in expressing their *expertise* and an algorithm based on collaborative filtering methods for ranking HPSs. Furthermore, by tagging SOA artifacts, human activities, and actions – defining

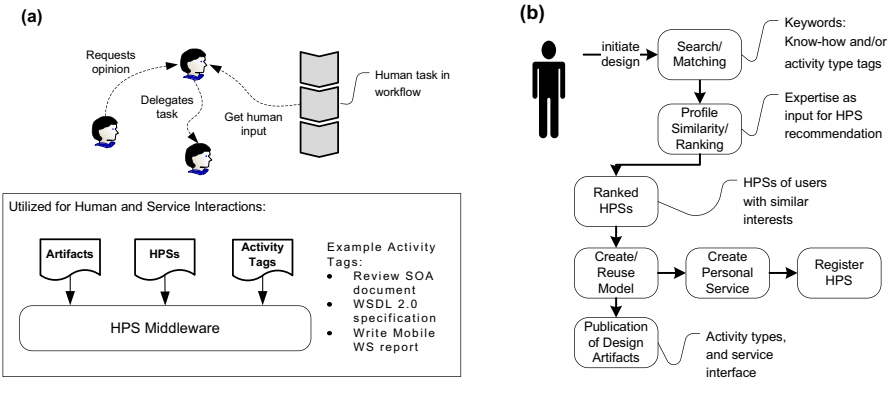


Fig. 1. (a) Typical collaboration scenario involving human- as well as human-service interactions. Tags are applied to various artifacts allowing for classification of services and user activities. **(b)** The design: users can reuse existing models or define new interfaces. Services are registered as personal services.

the type of collaborations in which an HPS is used – allows search based on user-defined keywords.

In Fig. 1, we show how the design is supported by utilizing tagged information and collaborative filtering methods. Tagging becomes increasingly important in today's collaborations allowing people to associate metadata to various artifacts — Web documents, links, messages, etc. (e.g., see [5] for usage patterns of tagging) and keyword-based search of user annotated content. Similarly, tags in the HPS framework are used to identify the *context* in which services and artifacts are used.

Generally speaking, tags are keywords/terms associated to information. In this work, we distinguish between tags assigned to *activities*, *services*, and *actions*. In the following we discuss how to utilize this metadata in the design process and detail the steps in Fig. 1.

1. Let's assume a user initiates the design process by deciding to define a service Fig. 1 (b). The common thing between the design of HPSs and, for example, creating service mashups is that both are user-defined services or processes. Like in most collaboration platforms, user-profiles managed by the HPS middleware contain information such as past/current activities and user preferences. Examples — in the computer science context — include "reviewer for a conference", "J2EE consultant", etc. We refer to such keywords available in the profile as activity information (e.g., predefined by the user).
2. The search for service artifacts is performed by matching the user's query vector against existing HPSs. A matching function takes service metadata as input, either automatically extracted keywords or, again tagged information. Hence, tagging information is not only used during collaboration, but also at design time. In contrast to above mentioned activity tags, service tags express how HPSs are used for collaborations (i.e., for actual interactions).
3. The next step is the ranking of HPSs. We compare a user's activities with the expertise of those HPSs matching the demanded set of keywords.
4. The user can create new models, publish related artifacts and type definitions using tools discussed in Section 4, or reuse existing HPS definitions. The model defines human activities which are mapped into *actions* (cf. [6]) performed using Web services.
5. Finally, *Personal Services* (the mapping of user profiles to services) are published in the HPS registry.

3 Overview HPS Framework

To enable HPS, we need a framework supporting the management of related artifacts, user profiles, and HPS interactions. The HPS framework has been developed for this purpose (cf. [2]). However, in this paper we focus on design related APIs and components which have been implemented on-top of the core framework. In the following, we provide a description of components in Fig. 2.

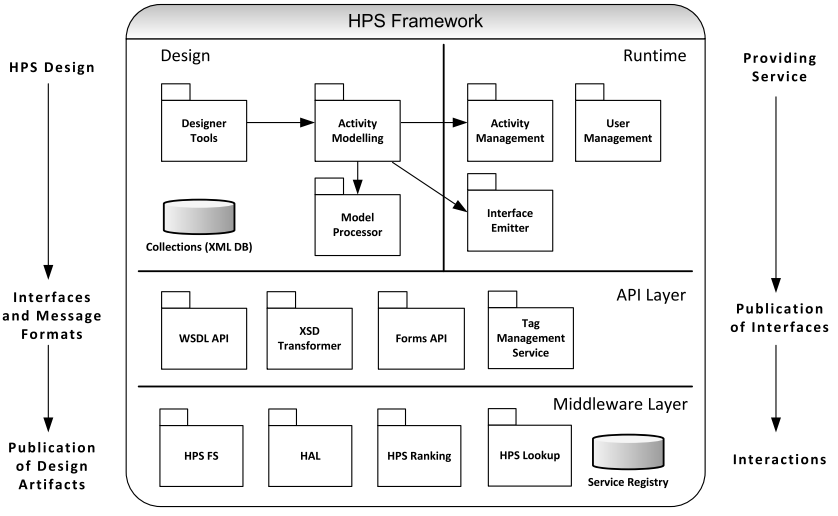


Fig. 2. HPS framework and architecture

The *API Layer* includes the core services for WSDL document generation (*WSDL API* service) which specify human activities and user specified interface elements (parameters and complex elements), the *Forms API* implementing support for XML Forms (XForms[2]), an *XSD Transformer* service utilizing the *Forms API* to automatically generate XForms based on XML schema definitions, for example, as defined in WSDL documents, and a *Tag Management* service associating tags with HPS artifacts (activities, actions, and WSDLs).

Design: Fig. 2 shows the design flow on the left side (top down) starting with *HPS Design* – designer tools such as a Web portal allowing users to create service interfaces in a simple manner.

1. *Interface and Message Formats:* the HPS framework provides tools to automatically translate high level specifications (e.g., activities and interface elements) into low level service descriptions without requiring the user to understand underlying technologies such as XML or WSDL.
2. *Publication of Design Artifacts:* artifacts such as message formats and activity type definitions are saved in XML collections.

Runtime: The *User Management* service holds user-related data such as profiles and contact details and is utilized by the *Activity Management* service. The *Interface Emitter* generates HPS interfaces depending on application scenario, for example, human interactions using HPS or human interactions in processes.

4 HPS Interface Transformation and Generation

The design process and methodology presented in this paper has to be supported by a set of tools and models. We start with the definition of the process, which

[2] W3C Markup Forms: http://www.w3.org/MarkUp/Forms/

allows users to define services without having to understand Web services technologies (depicted in Fig.3). Several papers focus on automatic GUI generation based on WSDL descriptions [7,8]. However, these works assume that the WSDL description of a service already exists and simply needs to be parsed and mapped into some GUI language/representation.

4.1 Design Process

We define a process allowing users to create an activity model serving as the input for automatic generation of service artifacts. The HPS framework provides *User Controls* and a corresponding *Meta Model* which enable people to design their *Activity Model* and *HPS Interfaces*. We discuss interface mappings and the meta model in Sec. 4.2.

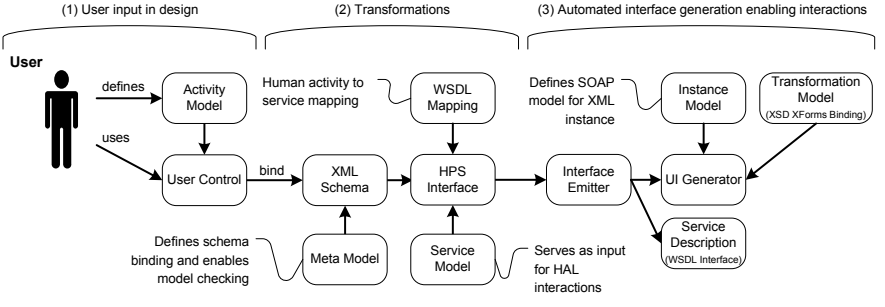


Fig. 3. Conceptual approach and interface design

Step 1 in the design: users define their *Activity Model* using controls (a simple example is shown in Fig. 5).

Step 2 comprises automatic transformations of the user's input into XML artifacts: i) invoke the `XSD Transformer` translating the activity model into XML schemes (i.e., *Schema Binding*) using definitions and constraints defined in the *Meta Model*. An example of such constraints and mappings is given in Table 1. Currently we express model mappings in XML schemes. ii) *HPS Interfaces* are created by associating activity types with the *Service Model*, defining how activity types (and human actions) are mapped into services definitions (WSDL). The mapping of an *HPS Interface* into WSDL is the binding of activity type definitions and actions with the HPS Access Layer (HAL, see [1]). At run-time, HAL acts as a proxy service dispatching requests by performing security checks, routing, message transformations (if needed), and persistency management of messages (i.e., saving request/response messages in XML collections).

Step 3 comprises the automated generation of interfaces at run-time. The *Interface Emitter* generates: i) interfaces allowing processes to interact with HPSs by generating WSDLs. Thus, HPSs may be included in (software) process by defining human activities in the process definition, which are enacted as HPS actions

Table 1. Excerpt interface mapping

Model & Binding	Description
XSD	`<xs:choice/>`
Form	`<xf:select1 ref="" appearance="$model"/>` with appearance as $model parameter ("full" or "compact")
Restriction/Model	`//*[@value='Choice']/prop[@name='type']/@value` and `<xs:choice minOccurs="$model" maxOccurs="$model">` with $model as parameters (minOccurs and maxOccurs being "1")

(interaction through HAL). GUIs are generated automatically by transforming WSDLs and XSDs into XML forms using the `Forms API`.

4.2 Interface Mappings

Meta Models define interface mappings of XML schemes into XForms. Table 1 shows an exemplary mapping of an XML type into an XForms representation, i.e., (1) *XSD* type, (2) *Form* model, and (3) mappings and *Restriction/Model* as defined in the meta-model.

The above mappings support transformations into a forms representations. We continue our discussion of HPS interfaces by showing a concrete XML example of a WSDL description. We start with type definitions in Listing 1.

Of course, no user input is needed for mapping activities onto services or to create WSDL descriptions. This is automatically performed by services in the framework. Listing 1 shows `GenericResource`, `ReviewRequest` type definitions.

Listing 1. Review-activity types example.

```
<xsd:schema targetNamespace="http://services.myhps.org/reviewservice">
  <xsd:complexType name="GenericResource">
    <xsd:sequence>
       <xsd:element name="Location" type="xsd:anyURI" />
       <xsd:element name="Expires" type="xsd:dateTime" />
    <xsd:sequence>
  </xsd:complexType>
  <xsd:element name="ReviewRequest" type="Request" />
  <xsd:complexType name="Request">
     <xsd:sequence>
       <xsd:element name="ReviewResource" type="GenericResource" />
       <xsd:element name="Comments" type="xsd:string" />
     </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="AckReviewRequest" type="xsd:string" />
  <xsd:element name="GetReviewReply" type="xsd:string" />
  <xsd:element name="ReviewReply" type="Reply" />
</xsd:schema>
```

Such definitions can be created using tools, as we shall discuss later. In this simplified example, the activity to be performed by a human is *review* comprising resources, the actual request, and the reply, which is complex XML data structure (abbreviated in this example).

Finally, Listing 2 shows an excerpt of a WSDL representing a review HPS.

Listing 2. HPS WSDL example.

```
<wsdl:message name="GetReview">
  <wsdl:part name="part1" element="ReviewRequest" />
</wsdl:message>
<wsdl:message name="AckReviewRequest">
  <wsdl:part name="part1" element="AckReviewRequest" />
</wsdl:message>
<wsdl:portType name="HPSReviewPortType">
 <wsdl:operation name="GetReview">
   <wsdl:input xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
         message="GetReview" wsaw:Action="urn:GetReview" >
   </wsdl:input>
   <wsdl:output message="AckReviewRequest" />
 </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="HALSOAPBinding" type="HPSReviewPortType">
 <soap:binding style="document"
       transport="http://schemas.xmlsoap.org/soap/http" />
</wsdl:binding>
```

Notice, `PortType` (i.e., the interface) for all interactions is the HPS access layer. At run-time, the access layer extracts and routes messages to the demanded HPS. Since every interaction is entirely asynchronous, interactions (session) identifier are automatically generated by the access layer (e.g., `AckReviewRequest`).

5 Recommendations for HPS Design

In the spirit of collaborative tagging systems, we perform ranking and recommendations based on tagged resources. An entry is a triple `(user, resource, {tags})`. We perform matching of relevant users who provide a particular HPS of interest. The matching function is based on applied tags. To obtain rankings, we measure the similarity of the user's preferences with activities and actions of matching users, i.e., those users already offering HPSs to find definitions of services that can be reused.

Algorithm 1 outlines our ranking approach. We assume several functions. For example, we calculate the similarity of preferences/activities as the Pearson correlation coefficient. Furthermore, we calculate the frequency of tags using additive smoothing, a simple yet effect method to account for missing/misplaced tags.

Algorithm 1. Recommendation algorithm comparing the preferences of a person v searching for existing HPSs and artifacts.

Require: U_{tag} is a set of users that applied the demanded set of tags. S is a vector holding the scores of services by users $\in U_{tag}$ with similar interests.
Require: We determine the correlation between users as:
$$correl(u_1, u_2) = \sum (u_1 - \overline{u_2})(u_1 - \overline{u_2})/(N * stdev(u_1)stdev(u_2))$$
1: **for** each user $u \in \mathcal{U}$ **do**
2: **for** each $tag \in AT$ **do**
3: /* Get the frequency for a specific tag used by u. */
4: $\Omega[tag] \leftarrow getFrequency(u, tag)$
5: **end for**
6: /* Assign the weight using the smoothing factor $0 < \gamma < 1$. */
7: $w_{(u,tag)} \leftarrow \frac{\Omega[tag]+\gamma}{\sum_{f_{tag} \in \Omega} f_{tag} + \gamma}$
8: /* If any of u's interactions/HPSs contain tag. */
9: Add u to U_{tag}
10: **end for**
11: **for** each user $u \in U_{tag}$ **do**
12: /* Calculate correlation between u and v. */
13: $\phi \leftarrow correl(w_u, w_v)$
14: $S[u] \leftarrow p(u) * score(\phi)$
15: /* $p(u)$ is personalization vector to assign additional preferences. */
16: **end for**
17: /* Sort preferred users by decreasing score and truncate S. */
18: return the top-k list of services by users

The input of Algorithm 1 is a query vector and the preferences of the user initiating the design process. By matching existing HPSs, we determine the initial set U_{tag}. We determine the score of a particular user by mapping the correlation factor ϕ into a linear function $score(\phi) = \frac{\phi+1}{2}$ since $(-1 \leq \phi \leq 1)$ would give us negative scores if there is no correlation between users. Additional personalization can be performed by assigning preferences (i.e., $p(u)$) for particular users. The output of the recommendation algorithm is a ranked list of services S.

6 Implementation

In this section we show the tools to manage HPSs, artifacts, etc. supporting lookup of services. The first tool illustrates how a user can search for existing services. The first screenshot in Fig. 4 shows the result of a user query. In this example we show the same HPS WSDL as discussed in previous sections (Sec. 4.2). If none of the existing HPSs fits the desired activities, users have the ability to create new artifacts and activities which are then mapped into HPSs.

The framework provides a set of designer tools which can be used to define control elements, options, and definitions of artifacts. A screenshot of a control is given in Fig. 5, allowing users to define complex data structures. This tool is used

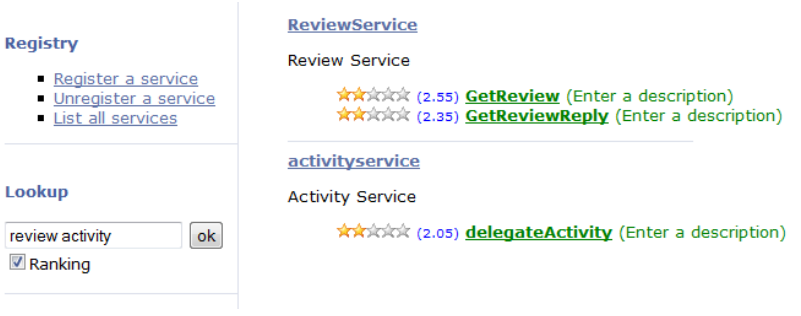


Fig. 4. Registry maintaining HPSs and service artifacts: users can search for services, which are displayed in a ranked list

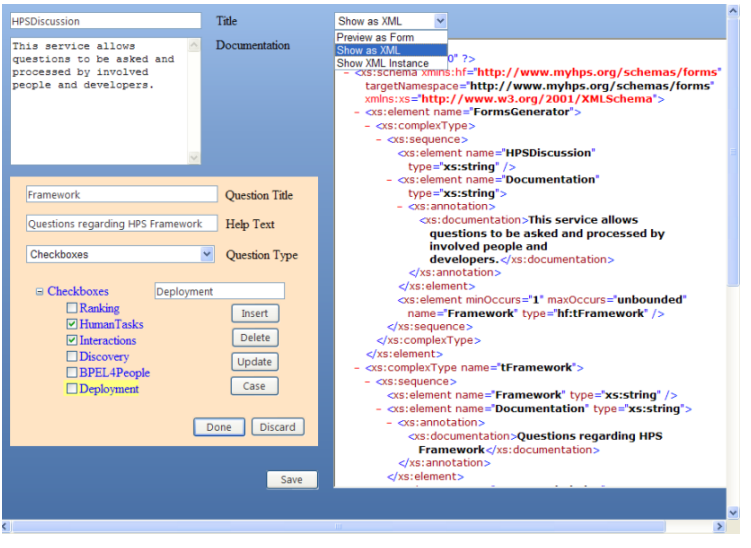


Fig. 5. Example control allowing users to define activities and complex data structures

in **Step 1** as defined by the design process in Sec. 4.1. The prototype version of this tools automatically translated user input into low-level XML artifacts such as schemes, instance documents (e.g., encapsulated in SOAP envelopes), WSDL descriptions and XML forms depending on the HPS use case. For example, interactions between humans in collaborations or using HPS in a process-centric scenario. `Show as XML` (for demo purposes) translates user defined elements into XML schemes, `Show as XML Instance` displays the associated XML schema instance document, and `Preview as Form` renders XForm presentation.

7 Related Work

The work presented in this paper focuses on a methodology and tools allowing humans to define service for various collaborations. We structure our discussion

into related work in the area of SOA, Web services, and process-centric collaboration. Second, we present related research in engineering methods in SOA. Third, we discuss tagging mechanisms and approaches for resource recommendations.

BPEL4People Task Model: BPEL4People defines human interactions in business processes via the human task specification [4]. A concrete implementation of BPEL4People as a service has been introduced in [9], but without supporting the design of services. In [10], the relation of various Web standards and resource patterns is discussed.

In contrast to above mentioned work, we describe a design approach allowing people to define services. While BPEL4People defines how developers can define human interactions in processes, related BPEL4People specifications do not describe how humans define services and how people manage interactions in SOA. The difference between the task model, such as defined by BPEL4People, and HPS is that tasks are usually defined for controlling interactions (e.g., start, end, deadline, etc.). HPS targets collaboration scenarios where users contribute their skills and expertise as services. Another point is that HPS reflects the composable nature of the Web, for example, reusing and composing services. However, we have not yet addressed compositions of HPSs.

Approaches for Interface Transformations and Generation: Our framework utilizes open standards such as WSDL, to describe HPS interfaces, and XForms to automatically generate user interfaces. GUI generation and mappings for WSDL has been presented in [7,8] and forms generators such as IBM's XML Forms Generator[3] for the Eclipse environment are available. In this paper we not only focus on generating GUIs based on WSDL, but also the mapping of human activities into HPS interfaces and generation of presentations (WSDL or XForms).

Tagging and Resource Recommendations: Recently, models for collaborative tagging have been presented [11] and personalized recommendations-based tagging models introduced in [12]. In [13], the authors present an evaluation of tag recommendations in folksonomies using collaborative filtering methods and an algorithm called FolkRank. Similarly, we follow a tag-based recommendation approach. However, we propose *activity tagging* to express expertise of users. Based on these tags, we implemented a collaborative filtering algorithm for recommendation of HPSs.

8 Conclusion and Future Work

The methodology and tools presented in this work offer support in the design of HPSs for users without having to implement code. The framework hosts a Web 2.0 portal, implemented in ASP.NET AJAX and C# APIs, allowing users to search for service artifacts (interfaces already provided by other users) and to design new services. The next steps include further implementation of XForm specifications and deployment of a rendering run-time on mobile devices. Also, we are working on composition aspects of HPSs and usability improvements of user tools.

[3] http://www.alphaworks.ibm.com/tech/xfg

References

1. Schall, D., Truong, H.L., Dustdar, S.: Unifying Human and Software Services in Web-Scale Collaborations. IEEE Internet Computing 12(3), 62–68 (2008)
2. Schall, D., Truong, H.L., Dustdar, S.: The Human-provided Services Framework. In: IEEE 2008 Conference on Enterprise Computing, E-Commerce and E-Services (EEE 2008), Crystal City, Washington, D.C., USA. IEEE Computer Society, Los Alamitos (2008)
3. Agrawal, A., Amend, M., Das, M., Ford, M., Keller, C., Kloppmann, M., König, D., Leymann, F., Müller, R., Pfau, G., Plösser, K., Rangaswamy, R., Rickayzen, A., Rowley, M., Schmidt, P., Trickovic, I., Yiu, A., Zeller, M.: WS-BPEL Extension for People (BPEL4People), Version 1.0 (2007)
4. Amend, M., Das, M., Ford, M., Keller, C., Kloppmann, M., König, D., Leymann, F., Müller, R., Pfau, G., Plösser, K., Rangaswamy, R., Rickayzen, A., Rowley, M., Schmidt, P., Trickovic, I., Yiu, A., Zeller, M.: Web Services Human Task (WS-HumanTask), Version 1.0 (2007)
5. Golder, S., Huberman, B.A.: Usage patterns of collaborative tagging systems. Journal of Information Science 32(2), 198–208 (2006)
6. Dustdar, S.: Caramba a process-aware collaboration system supporting ad hoc and collaborative processes in virtual teams. Distrib. Parallel Databases 15(1), 45–66 (2004)
7. Song, K., Lee, K.H.: An Automated Generation of XForms Interfaces for Web Services. Web Services. In: Song, K., Lee, K.H. (eds.) IEEE International Conference on ICWS 2007, pp. 856–863, July 9-13 (2007)
8. Kassoff, M., Kato, D., Mohsin, W.: Creating GUIs for Web Services. IEEE Internet Computing 07(5), 66–73 (2003)
9. Thomas, J., Paci, F., Bertino, E., Eugster, P.: User Tasks and Access Control over Web Services. In: Int. conf. on Web Services (ICWS 2007), Salt Lake City, USA, pp. 60–69. IEEE Computer Society, Los Alamitos (2007)
10. Russell, N., Van Der Aalst, W.M.P.: Evaluation of the bpel4people and ws-humantask extensions to ws-bpel 2.0 using the workflow resource patterns. Technical report, BPM Center Brisbane/Eindhoven (2007)
11. Cattuto, C., Loreto, V., Pietronero, L.: Semiotic dynamics and collaborative tagging. PNAS 104(5), 1461–1464 (2007)
12. Byde, A., Wan, H., Cayzer, S.: Personalized tag recommendations via tagging and content-based similarity metrics. In: Proceedings of the International Conference on Weblogs and Social Media (2007)
13. Jäschke, R., Marinho, L.B., Hotho, A., Schmidt-Thieme, L., Stumme, G.: Tag recommendations in folksonomies. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) PKDD 2007. LNCS, vol. 4702, pp. 506–514. Springer, Heidelberg (2007)

Model Transformations to Leverage Service Networks*

Marina Bitsaki[1], Olha Danylevych[2], Willem-Jan A.M. van den Heuvel[3], George D. Koutras[1], Frank Leymann[2], Michele Mancioppi[3], Christos N. Nikolaou[1], and Mike P. Papazoglou[3]

[1] Computer Science Department, University of Crete, Greece
{bitsaki,koutras,nikolau}@tsl.gr
[2] Institute of Architecture of Application Systems, University of Stuttgart, Germany
{olha.danylevych,leymann}@iaas.uni-stuttgart.de
[3] INFOLAB, Dept. of Information Systems and Management, Tilburg University, The Netherlands
{wjheuvel,m..mancioppi,mikep}@uvt.nl

Abstract. The Internet has catered for the transformation of traditional "stovepiped" service companies into global service networks fostering co-production of value to more effectively and efficiently satisfy the ever-growing demands of mundane customers. The catalyst of this change is the happenstance of Service Oriented Computing, which provides a natural distributed computing technology paradigm for implementing and evolving such highly distributed networks of autonomous trading partners with coordinate and cooperative actions. However, how to faithfully (re-)map service networks to business processes and service realizations and vice-versa is still partly terra incognita.

In this paper, we introduce a semi-automatic model transformation approach for creating the abstract business processes that take place between trading partners from models of service networks, assuming limited human-involvement focused on selecting reusable transformation patterns. This approach is explored and validated using a realistic case study reflecting best practices in the telecommunications industry.

1 Introduction

The services industry has become the leading contributor to business activities in developed economies, encompassing sectors such as logistics, education, publishing, finance, healthcare, telecom and government. The digitally networked service economy, driven by distributed computing technologies such as *Service Oriented Computing* (SOC) [1], is believed to revolutionize the way in which these companies conduct business, enabling exiting new business models such as service networks.

Service Networks. (SNs) [2,3] leverage end-to-end service interactions between network partners that embody a succession of business processes typically cutting

* The research leading to these results has received funding from the European Community's Seventh Framework Programme under the Network of Excellence S-Cube - Grant Agreement n° 215483.

G. Feuerlicht and W. Lamersdorf (Eds.): ICSOC 2008, LNCS 5472, pp. 103–117, 2009.

across organizational boundaries and spanning various geographical locations. Service networks sequence service activities with the flow definitions of business process modelf into end-to-end service constellations, assign work items to the appropriate human actors or groups, and ensure that both human- and systems-based activities are performed within agreed-upon timeframes and QoS criteria.

SOC is touted as the de-facto distributed enterprise-computing technology for developing and evolving SNs. In a SOC-based environment, business processes can be implemented as networks of choreographed services between- and orchestrated services within- network partners, relying on global standards including BPEL and WSDL. *Business Process Management* (BPM) [4] is a natural supplement to SOC through which business activities can be monitored and measured across business processes and services, while maximizing business value in service networks. In a nutshell, BPM has been evolved into a comprehensive lifecycle model that encompasses (graphical) process analysis & design, process execution, and process monitoring and reporting capabilities.

SOC-based design & development in tandem with BPM-based management of SNs should be grounded on a methodology offering a consistent body of methods, notations and tools. As a first fundamental step, [2] proposed the *Service Network Notation* (SNN) as a novel modeling language enabling quantitative economic analysis of SNs to ascertain the optimal constellation of collaborative economic agents resulting in maximum economic value. Service networks described as SNN models can be analyzed to optimize the value generated in the network using financial metrics like cost, revenues and customer satisfaction. Further evolution of the SNN notation and its related analysis techniques will concern the simulation of service networks to discover value anomalies (e.g. services or partnerships that do not produce as much value as expected) before the actual services are realized and deployed, and to perform different types of "what-if" analysis, such as study the changes of value flows upon replacements of services and partners, broadening or shrinking of the market (i.e. more participants join or live the service networks). This paper pursues new steps towards the alignment of service networks and the underpinning business processes to realize such a comprehensive service network analysis, development and management methodology.

The development of business processes and services are already part of the current state of the art of subsequently BPM and SOC, and is well understood [5]. Thus, we will not any further consider them in this work. Also, mapping business processes and services has been extensively scrutinized in the field of *Model Driven Engineering* (MDE) [6] and *Model Driven Architecture* (MDA) [7]. MDA is an effort of the Object Management Group providing the foundations and promoting the generation of programming code from models. Model transformations are used to generate new models (e.g. source code in some programming language) from other models (e.g. UML2 Class Diagrams) using repeatable (automatic) processes expressed as rules [8].

Similarly to their applications to software engineering, MDE and MDA harbor huge potential benefits for BPM and service networks. In BPM, a wide variety

of transformations have been devised to facilitate the generation, for instance, of executable business processes from abstract ones (such as, but not limited to, [9,10]). In the ambit of service networks, one of the links currently missing is how to exploit the information about value flows among participants contained in SNN models to streamline the generation of those business processes that are the backbone of the service networks, and vice-versa how to extract service network representations from existing abstract business processes. In this paper we bridge this gap by introducing a transformation approach for constructing business processes in from models of service networks and the other way around, which is accelerated through the usage of process interaction patterns that can be injected during the transformation processes. This results into an approach that is one of continuous (re-)design, scoping, refinement and adjustment of service network- and abstract business process models.

The paper is organized as follows: Section 2 outlines the SN4BPM architecture on which our transformation is grounded. Section 3 then introduces a realistic running example of a service network for customer- and network fault handling. Section 4 describes a staged transformation method to map SNs into business processes, after which Section 5 elaborates the transformation mechanisms in detail. Finally, Section 6 concludes the paper with conclusions and directions future work.

2 The SN4BPM Architecture

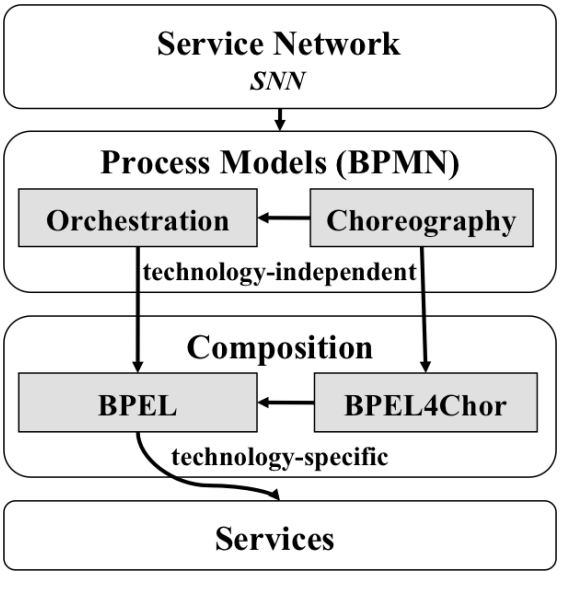


Fig. 1. The SN4BPM stack

The SN4BPM (Fig. 1) entails a stratified architecture that serves as the foundation for realizing business processes in service networks. This architecture unifies the BPM and SOC standards stacks for realizing service networks, fostering a clean separation of concerns among the devising of the strategies, partnerships and their effects, the abstract business processes realizing them, the executable business processes and the underpinning IT infrastructure.

At the top layer of this architecture, the *Service Network* layer encompasses service network models that serve as the basis of analyzing, simulating and optimizing networked constellations of business partners, each of which contributes to the network processes, while adding value.

The *Process Models* layer deals with modeling abstract business processes with languages such as the *Business Process Modeling Notation* (BPMN). Abstract business process models are implementation agnostic, and harness processes as orchestrations of services, each of which realizes an activity and is executed according to a control flow that may be private or public to the partners in the SN. Abstract business processes thus omit implementation details that are necessary to the execution of the business processes (e.g. the endpoint of the services to interact with). Abstract business processes can be partitioned into *business process fragments* that group independent and cohesive subsets of interactions among the participants. For example, an abstract business process *CustomerProblemHandling* in a telecommunications service network can be partitioned into two cohesive business process fragments, viz., *ProblemDiagnostics* and *ProblemFixing*.

In particular, abstract processes are realized as executable processes in the *Composition* layer, where they can be rendered as *choreographies*, which provide a global view on the (inter- and intra-organizational) multiparty collaborations focusing on the message-based communication among partners, or *orchestrations*, which specify and connect into executable workflows the activities performed and the message exchanges performed by a participant or a service. Executable business processes are technology-dependent, and are usually modeled using languages such as the *Web Services Business Process Execution Language* (WS-BPEL) and *BPEL4Chor* [11], respectively focusing on orchestrations and choreographies. The Services layer provides the set of discrete services available in the service network, relying on open standards based message backbone, enabled by SOC infrastructural plumbing technologies such as an *Enterprise Service Bus* [12].

3 The Service Network Notation

The transformation approach will be illustrated and explored with a simple and realistic running example concerning a service network for resolving resource and service problems that are reported by Telco clients, e.g. connection problems, in a telecommunications service network comprised of consumers, intermediaries, telco service providers and suppliers. This case study is based on a description of standard, end-to-end business processes in the *eTOM Business Process Framework* [13]. For reasons of understandability we will now briefly explain the basic concepts in SNN.

A comprehensive overview of the SNN notation and its meta-model are provided in [2]. Fig. 2 depicts our running example as an SNN model. It focuses on two essential processes in fault resolution, the *Customer Fault Resolution* and the *Network Fault Resolution* business processes. The Customer Fault Resolution process conceptualizes the customers procedure for reporting a fault to a Customer Service Representative (CSR): after the reception of a trouble ticket, the CSR delegates the resolution job to a Field Agent, after which the Field Agent intervenes at the Customers site to solve the issue. When the issue is solved, the Customer pays for the intervention.

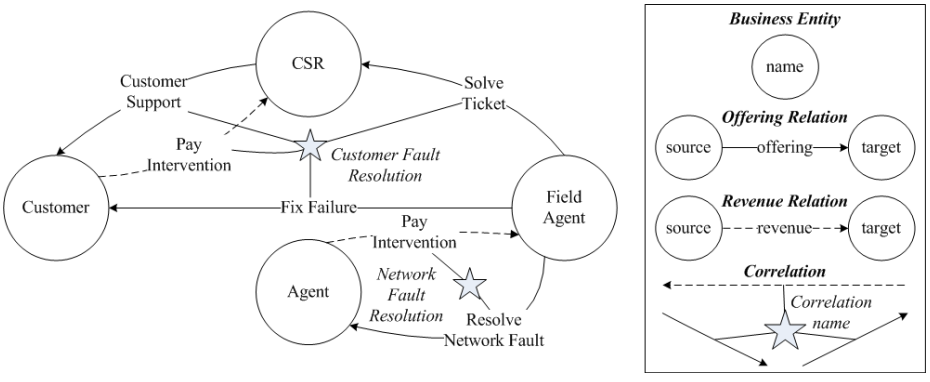


Fig. 2. An SNN model representing the eTOM example

The SNN model captures this scenario among the three business entities, *Customer*, *CSR* and *Field Agent* as follows. The provisioning of the resolution service from the CSR to the Customer is represented as a directed arrow labeled as the *Customer Support* offering. The Field Agent supplies the Customer with the *Fix Failure* service and the CSR with the *Solve Ticket* service. The Customer pays for the intervention by generating revenue for the CSR, which is modeled with the dotted directed arrow labeled *Pay Intervention* pointing to the CSR. The *Customer Support*, *Solve Ticket*, *Fix Failure* and *Pay Intervention* relations are correlated (the star-like symbol connecting them) because they all take place within the context a single business process fragment, called *Customer Fault Resolution*, with a clearly demarcated start and end.

4 Transformations in the Enhanced BPM Lifecycle

The development practices and activities for developing service-enabled processes in SNs are organized and integrated by the *enhanced BPM lifecycle* introduced in [2], which is depicted in Fig. 3.

This lifecycle extends the traditional BPM lifecycle by introducing a phase called *Rationalization* that deals with the design, optimization and simulation

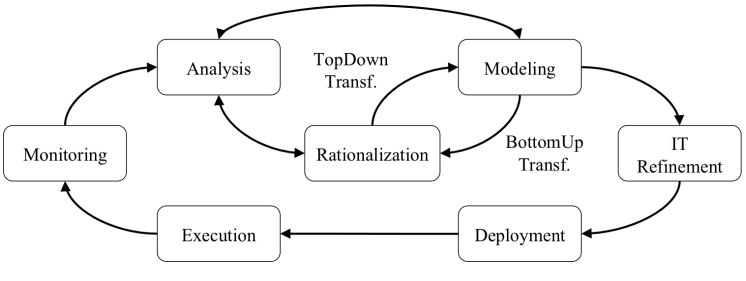


Fig. 3. The enhanced BPM lifecycle

of SNN models. The *Analysis* phase elicits and collects business process requirements, and resolves requirement inconsistencies and incompleteness. The *Modeling* phase addresses design, maintenance and evolution of abstract business process models. The *IT refinement* phase centers on realizing executable process models, which are then deployed on the IT infrastructure during the *Deployment* phase. The *Execution* phase enables the execution of processes, and generating execution trails which are used in the *Monitoring* phase to adapt particular process instances while still running, detect trends and patterns in the current usage of the processes, keep track of the overall state of the system, etc.

Transformations are the mortar that bind the new elements of the enhanced BPM lifecycle with the well-established practices of the standard BPM lifecycle, enabling the analyst to move from the Modeling phase to the Rationalization phase and vice-versa as shown in Fig. 3. The connections between the Analysis and Rationalization phases are based on one direction on modeling SNN models that capture the requirements specified on the Analysis phase, and on the other on extracting new requirements from optimized SNN models. The transformations that are scrutinized in this paper foster in the lifecycle the bi-directional synchronization of artifacts designed during the Modeling- and Rationalization phases, viz. the SNN model and the abstract business process model that are logically associated through the stratified SN4BPM architecture.

5 The Transformation Approach

Enhanced business process management is facilitated when the business analyst may easily progress from one phase to the next and back to iteratively develop service-enabled processes for new service networks, and to incrementally deal with changes in existing ones.While not incremental and iterative, the transformation approach here proposed is an initial step towards this vision. The *TopDown* and *BottomUp* transformations, named after the relative directions in traversing the SN4BPM stack of Fig. 1, respectively allow to map service network models (belonging at the Service Networks layer) into abstract business process model (at Process Models layer in the stack) and vice-versa through a multi-step approach Mapping SNN models (belonging at the Service Networks layer) to abstract business process models (at Process Models layer in the stack).

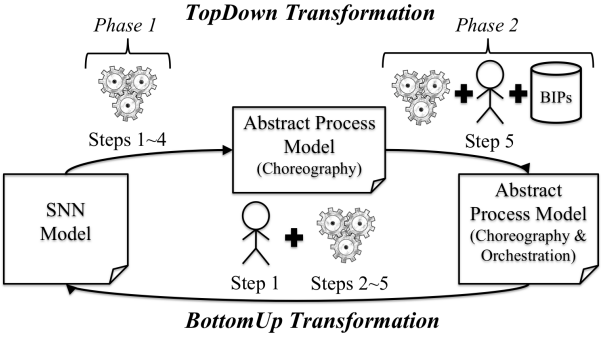


Fig. 4. Overview of the transformation approach

Fig. 4 provides an overview of the basic workings of the transformation approach indicating whether steps are completely automatic (gears), or that require some decisions taken by an analyst (sticky figure), or involve *Business Interactions Patterns* (BIPs). A BIP is a generic and reusable template of a business process fragment that can be applied to concepts in the SNN model (e.g. correlation). In particular, a BIP summarizes roles played by participants in a business fragment, workflows structuring the activities performed by the roles, and message-based interactions that occur among the different roles.

As depicted in this Fig. 4, the BottomUp transformation is composed of five steps, one requiring human intervention and the other automatic, that produce a well-formed SNN model from a BPMN model. The first step requires the analyst to label the message-flows at BPMN level that represent revenue and offering relations between participants at SNN level. From then on, the BottomUp transformation is fully automated. The TopDown transformation creates a BPMN model from the information embedded in a SNN model. The transformation is divided in two main phases, the first required and the second optional. The steps from 1 to 4 (phase 1) are completely automatic and start from a SNN model to result in a BPMN model that describes pools, lanes that divide the pool in independent business process fragment, sub-processes captured in the lanes, and also defines interrelationships between sub-processes through message-flows that mirror interactions at SNN level. Step 5 (phase 2) is semi-automatic, as it requires human involvement for the selection of the BIPs to be applied.

Both transformations are based on the mapping between the SNN and BPMN meta-models that is presented in Section 5.1. The business interaction patterns are examined more in depth in Section 5.2. The BottomUp and TopDown transformations are respectively described in detail in Section 5.3 and Section 5.4.

5.1 Model Mappings for SNN and BPMN Model Transformation

Both the TopDown and BottomUp transformations are defined on the basis of the mapping between the SNN and the BPMN[1] meta-models presented in Fig. 5 (see the bold bi-directional arrows in this figure).

Each SNN *Service Network* corresponds to a BPMN *process*. *Participants* in SNN models are mapped to *pools* in BPMN. SNN correlations group interactions among participant in different business process fragments. A participant involved in interactions spread over multiple business process fragments has multiple lanes in its pool, one per fragment. For instance, if a participant takes part in interactions that are divided into three different business process fragments, its respective pool will contain three lanes. In BPMN, for all practical purposes, pools with a single lane and pools without lane are equivalent. Pools and lanes contain *workflows* (i.e. *activities* connected by control flow constructs). *Sub-processes* are special activities that abstract entire workflows. Workflows and sub-processes can be recursively nested into each other. Activities communicate

[1] Extrapolated from the BPMN meta-model published on WSPER.ORG based on [14], and available at: `http://www.wsper.org/bpmn10.html`

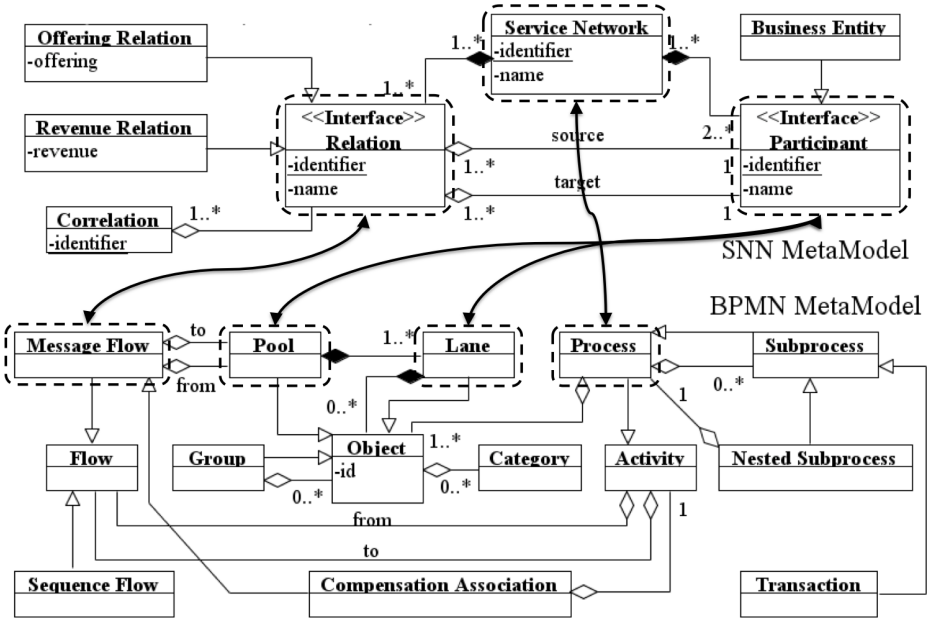


Fig. 5. The mappings between the SNN and BPMN meta-models

with other activities in different pools through *message-flows*, which represent message exchanges. Both *offering-* and *revenue relations* in SNN are mapped to message-flows in BPMN.

These mapping are based on two critical assumptions about the structure of the BPMN. Interactions among participants in the SNN are represented by message-flows in BPMN (i.e. the participants in the service network carry out their interactions over message-based conversations). Secondly, each pool only comprises business logic (workflows) of one particular business fragment. Workflows in different lanes within the same pool are independent, i.e. they are not connected through control flows. The first assumption allows the BottomUp transformation to derive the offering and revenue relations to be represented in the resulting service networks. The second assumption enables the BottomUp transformations to cluster message exchanges between participants in different abstract business processes as correlations in the SNN model.

5.2 Business Interaction Patterns under the Lens

The Business Interaction Patterns play a central role in creating via the Top-Down transformation BPMN models that are immediately usable in a BPM lifecycle to, for instance, automate the generation of executable BPEL processes as proposed in [9]. SNN models describe which interactions take place between the participants, but not how these interactions are structured (e.g. in terms of message exchanges and activities performed by the involved participants). It is not possible for an automatic transformation to "guess" how participants will

communicate with each other to carry out the interactions described at SNN level, and this is mirrored by the coarse granularity of the intermediate BPMN models resulting from the first four steps of the TopDown transformation. BIPs are meant to improve the level of detail of the business processes resulting from the transformations by providing a structure (based on message-exchange communication) for the abstract interactions at SNN level in the shape of business process fragments that are selected by humans and are automatically "plugged" in the BPMN models resulting from the first transformation phase.

BIPs are based on existing business process standards in industry Such as RosettaNet *Partner Interface Process* (PIPs) [15], which are widely adopted reference models for standard, multi-party collaborative business process models. In the remainder of this paper we assume that BIPs are modeled as context-independent BPMN models where each role is represented by a pool. Each pool captures a well-structured sequence of internal processing steps; we refer to this as a workflow. Workflows in different pools are logically interconnected via message-flows.

Fig. 6 presents an example of BIPs that we have developed, called "On Behalf Of" BIP. It defines a BPMN template for a chain of value-relationships between multiple partners participating in a correlation within the service network, and is computed as a transitive relation between a *Provider* and a *Customer* comprising a cohesive path of message-exchanges that involve the sub-contractor (*Service Facilitator*) as "man in the middle". A series of delegated service offerings are subsequently traversed during the execution of a particular business process fragment, e.g. the *CustomerFaultResolution* process. Note that for reasons of brevity, we concentrate on a simple "On Behalf Of" correlation involving three participants; however, in practice we have already encountered correlations involving larger chains for which we have designed more complicated BPMN templates (e.g. for the automotive repair service scenario introduced in [3]).

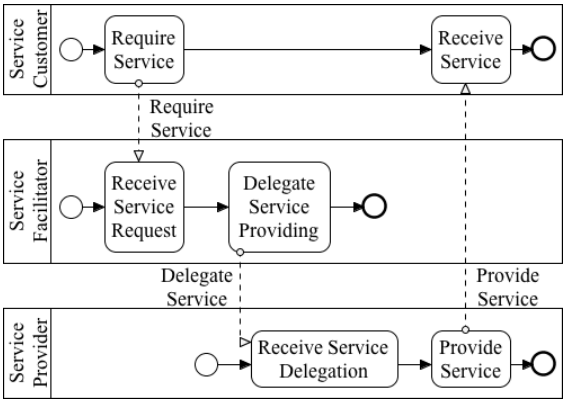


Fig. 6. The "On Behalf Of" Business Interaction Pattern

5.3 The BottomUp Transformation: Extracting SNN Models from Abstract Process Models

The BottomUp transformation extracts a service network model from an abstract business process. As explained in Section 5.1, the correspondences between business entities in SNN and participants in BPMN are rather straightforward.

On the other hand, it is much harder to extrapolate from an abstract business process, in which the participants interact over message exchanges, what kind of revenue and offering relations occur. The first step of the BottomUp transformation, requiring human intervention, tackles this issue by having an analyst label the message-flows in the source BPMN model that represent interactions between participants that have to be represented at SNN level in the shape of revenue or offering relations. This approach relies on the assumption that each relation at SNN level is represented by (at least) one message flow in the source BPMN model.

The BottomUp transformation produces an SNN model from a *source* BPMN model using the following five steps (see also Fig. 4):

1. *Label message-flows that represent revenue- or offering-based interactions:* the analyst labels the message-flows that represent offering or revenue relations at SNN level as shown in Fig. 7. For instance, the message-flow "Solve Ticket" represents an offering relation with the Field Agent as source and the CSR as target, while "Pay Intervention" represents a revenue relation from the Customer to the CSR.
2. *Collapse sub-processes in the source BPMN model: expanded* sub-processes (i.e. sub-processes that show their internal workflow) are transformed in *collapsed* sub-processes as explained in [14].
3. *Create the participants:* for each pool in the source, create a participant in the SNN model, and name the participant after the pool.
4. *Create offering and revenue relations:* for each message-flow connecting an activity in the pool of participant $\mathcal{A}$ with an activity in the pool of participant $\mathcal{B}$, do as follows:
   (a) If the a group of message flows is labeled as a revenue relation, then create a new revenue relation in the SNN model connecting participant $\mathcal{A}$ to participant $\mathcal{B}$ and using the name of the message-flow as the revenue offering associated with the newly created revenue relation $\mathcal{R}$. Participant $\mathcal{A}$ and $\mathcal{B}$ are respectively source and target of $\mathcal{R}$.
   (b) If the message flow is labeled as an offering relation, then create a new offering relation in the SNN model connecting participant $\mathcal{A}$ to participant $\mathcal{B}$ and using the name of the message-flow as the offering associated with the newly created offering relation $\mathcal{O}$. Participant $\mathcal{A}$ and $\mathcal{B}$ are respectively source and target of $\mathcal{O}$.
   (c) If neither Step 4a nor Step 4b apply, then ignore the message-flow.
5. *Create correlations:* group the message-flows in the source BPMN model according to the business process fragments they belong to. This is obtained by:
   (a) Grouping the workflows in the participants lanes in *workflow-groups*. Two workflows belong to the same workflow-group if they are connected by a message-flow. Namely, a workflow-group is the *transitive set*[2] of workflows that are connected by a message-flow.

[2] In other words, conversations at SNN level are identified in Step 5 by calculating the transitive sets of the message-flows on the basis of the workflows they connect, and creating a new correlation for every transitive set, connecting all the revenue and offering relations originated by the message-flows in that transitive set.

(b) Grouping message-flows in *message-flow-groups*. Two message-flows belong to the same message-flow-groups if they originate from or end in workflows grouped in the same workflow-group. Alternatively, a message-flow-group is the transitive set of message-flows that connect workflows in the same workflow-group.
(c) For each message-flow-group, create a correlation connecting all the offering- and revenue relations that have been created starting from the message-flows in the message-flow-group.

Consider the BPMN model in Fig. 7 that models the key abstract business processes in our running example. The first step in the transformation is to label the message-flows as revenue- or offering-relations. The second step is to collapse the sub-processes. After these first two steps, the resulting BPMN model looks like the one presented in Fig. 7. Fig. 8 exemplifies the three remaining steps in the BottomUp transformation. The third step of the transformation creates the participants in the SNN model (result shown in Fig. 8, Step 3). The revenue and offering relations in the SNN model are created in the fourth step (outcome presented in Fig. 8, step 4). Finally, the correlations are added to the SNN model during the fifth step (result in Fig. 8, step 5). The workflow groups are two: the workflows in the "Network Fault Resolution" sub-processes in the Agent and Field Agent pools (they are connected by the message-flows "Resolve Network Fault" and "Pay Intervention"), and the workflows named "Customer Fault Resolution" in the pools Field Agent, CSR and Customer (transitively connected by the "Customer Support", "Solve Ticket", "Fix Failure" and "Pay Intervention" message-flows).

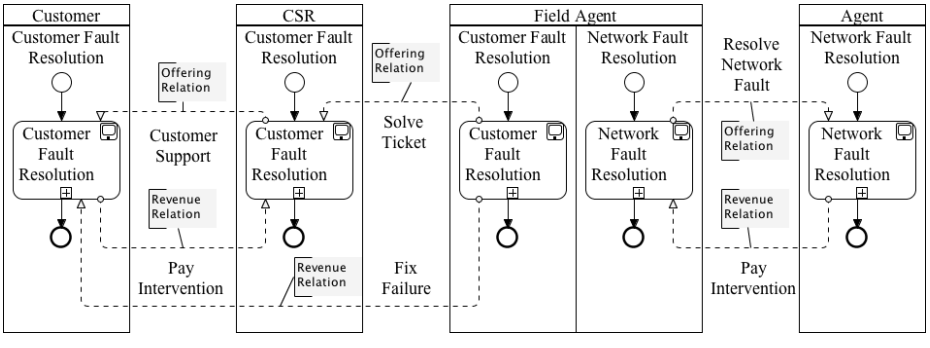


Fig. 7. A BPMN model based on the eTOM example presented in Section 3

5.4 The TopDown Transformation: Creating Abstract Process Models from SNN Model

In the following we introduce the TopDown transformation to refine a SNN into a BPMN model as a two-staged process.

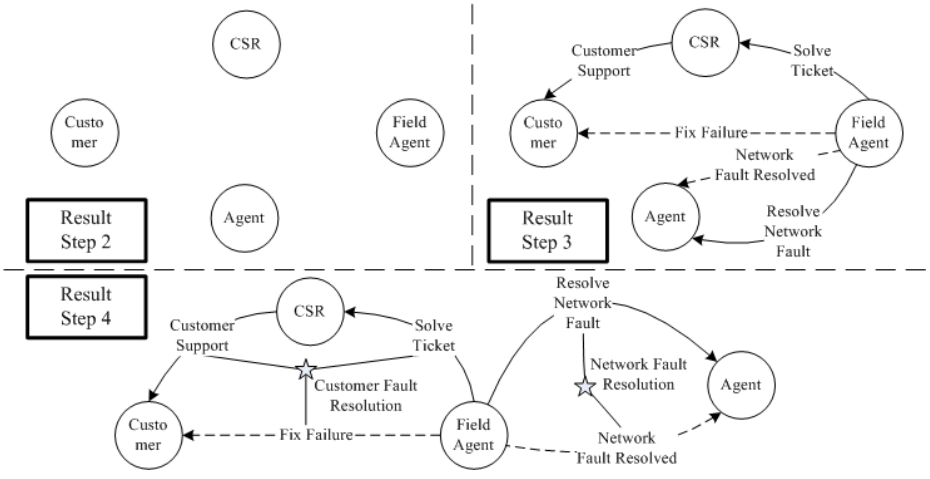


Fig. 8. The results of Step 3, Step 4 and Step 5 of the BottomUp transf. on the BPMN model in Fig. 7

PHASE 1 (Required): Produce an Abstract Business Process Choreography Model. This abstract business process model is rendered in BPMN, and defines the process choreography tracking the globally visible message flows between network partners. It is automatically generated in the following manner:

1. *Create the pools:* for each participant in the SNN model, create a pool in the BPMN model with the same name.
2. *Create the lanes and simple workflows:* for each correlation in the SNN model a participant is involved in, create a lane (named after the correlation) in the participants respective pool. If a relation in the SNN model does not belong to any correlation, it is treated as it belonged to a correlation comprising only itself. Each lane thus created this way is filled with a workflow made of a start event, a sub-process named as the correlation, and an end event sequentially connected in this order.
3. *Create revenue message-flows:* for each revenue relation $\mathcal{R}$ in the correlation $\mathcal{C}$ connecting the source participant $\mathcal{A}$ and the target participant $\mathcal{B}$, create a message-flow from the sub-process in the lane $\mathcal{C}$ of the pool $\mathcal{A}$ to the sub-process in the lane named $\mathcal{C}$ of the pool $\mathcal{B}$. Let the revenue of the relation $\mathcal{R}$ be revenue. The newly created message-flow is labeled as: "[Revenue] revenue".
4. *Create offering message-flows:* for each offering relation $\mathcal{O}$ in the correlation $\mathcal{C}$ connecting the source participant $\mathcal{A}$ and the target participant $\mathcal{B}$, create a message-flow from the sub-process in the lane $\mathcal{C}$ of the pool $\mathcal{A}$ to the sub-process in the lane named $\mathcal{C}$ of the pool $\mathcal{B}$. Let the offering associated to the relation $\mathcal{O}$ be offering. The newly created message-flow is labeled as: "[Offering] offering".

PHASE 2 (Optional): Produce an Extended Abstract Business Process Model. The resulting model not only defines the process choreography,

but also captures the private workflows of network partners. This phase is optional if the level of detail in the BPMN model resulting from the first phase is deemed insufficient, and thus requires further refinement through the application of one of more business interaction patterns. The phase is composed of the following semi-automatic step:

1. *Apply BIPs:* the application of a BIP requires the user select a correlation (*selected correlation*) in the SNN model (again, relations not involved in any correlation are treated as if they belonged in a correlation comprising only themselves). The participants that are source or target of offering and revenue relations that are comprised in the selected correlation are called involved participants. The user provides a mapping from the roles in the BIP to the *involved participants*. The BIP must define as many roles as the involved participants. For each involved participant, the workflow in the mapped roles pool is copied inside the lane created at Step 2 for the selected correlation in the pool corresponding to that participant. Finally, all the message-flows in the BIP are automatically copied into the BPMN model and connected to the same activities as they are in the BIP.

Fig. 7 visualizes the result of applying the first four steps to the SNN model in Fig. 2. The Field Agent participant is transformed in the Field Agent pool (Step 1). Since the Field Agent participant in the SNN model has offering and revenue relations grouped in different correlations ("Customer Fault Resolution" and "Network Fault Resolution"), the Field Agent pool has two lanes, one per correlation (Step 2). The revenue relation Resolve Network Fault in the SNN model is transformed into a message-flow, labeled "Resolve Network Fault"

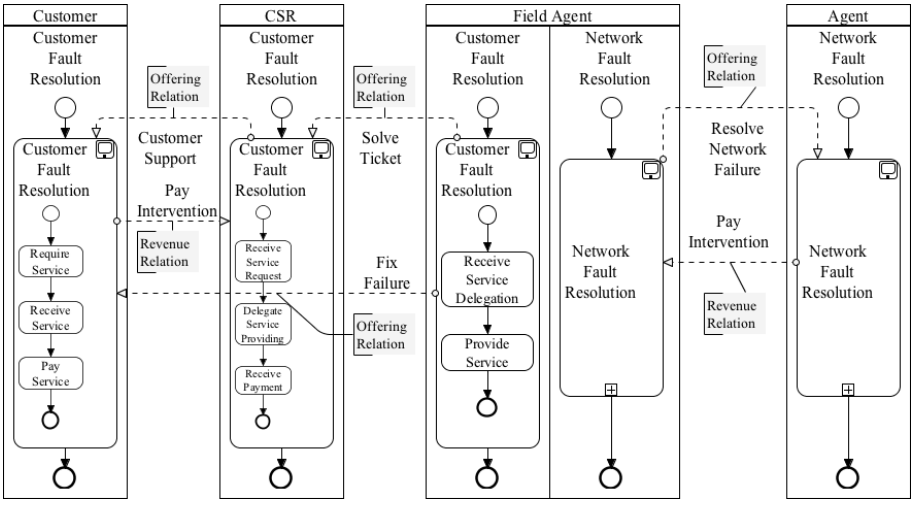


Fig. 9. Result of application of the "On Behalf Of" BIPs to the business process fragment "Customer Fault Resolution"

(Step 3) which connects the two sub-processes in the lanes of the Agent and Field Agent pools that are created because of the correlation.

Fig. 9 shows the results of the application of the fifth transformation step ("Apply BIPs") to our running examples. At the left hand side in the result of applying at the fifth step the "Provide Service Within Deadline" BIP to the correlation "Network Fault Resolution" by mapping the role "Service Provider" to the Field Agent, and "Service Requestor" to the Agent, is exemplified. At the right hand side of this figure, it demonstrates the application of the "On Behalf Of" BIP to the BPMN model in Fig. 7 by respectively mapping the "Service Requestor", "Service Facilitator" and "Service Provider" roles to the Customer, CSR and Field Agent pools.

6 Future Work and Conclusions

The service industry, the leading contributor to developed economies, is quickly transitioning towards the digital networked economy that is leveraged through distributed computing technologies including Service Oriented Computing. In conjunction with its natural complement Business Process Management, SOC is touted as the ideal paradigm to develop, evolve and manage sophisticated service networks that enact successions of automated end-to-end business processes that traverse several enterprises and geographical locations. However, this vision is far from a reality and many organizations are still fixated on orchestration of internal processes, witnessing the popularity of languages such as BPEL.

Service networks promise to effectively leverage and bridge between business-like requirements such as value and revenues, and the IT enactment through Service Oriented Architecture and Business Process Management. Service networks have recently catered a wide interest, which resulted, among other proposals, in the Service Network Notation that describes the interactions among participants in a service network in lieu of offering- and revenue relations.

In this paper, we have proposed and explored a semi-automatic approach for constructing business processes in service networks, or redefining service networks after changes to business processes. The proposed transformation approach is grounded on a series of mappings between the meta-models of SNN and BPMN models, and is formalized through procedural transformation algorithms. This approach is almost completely automated, and assumes restricted involvement of human experts restricted to the selection of the business interaction patterns that best capture recurrent skeletons of interactions between and within processes partners in the service network, and to the labeling in the business processes of message exchanges that need to be represented at service network level.

The results presented in this paper are core results in nature. Extensions and refinements are needed in various directions. Firstly, we intend to further elaborate the transformation approach to make it incremental and iterative, and to improve the BottomUp transformation to use pattern recognition mechanisms to automatically extract the revenue and offering relations at SNN level by applying "backwards" the BIPs (i.e. recognize process fragments that fit BIPs and generate the corresponding relations at SNN level). In addition, we intend to further

explore and elaborate the transformation approach in several real case studies. Thirdly, we wish to further extend the library of business process interaction patterns. The BIP library, currently populated with a handful of patterns, will be extended with existing patterns that can be easily extracted from industrial reference models, standard protocols and industrial best practices. Moreover, we intend to investigate more complex transformation scenarios where multiple business interaction patterns occur in the same business process. Lastly, we are in the process of implementing the transformations in the *Value Network Tool* (http://vnt.tsl.gr/), the integrated development environment that supports the design of SNN models.

References

1. Papazoglou, M.P.: Service-oriented computing: Concepts, characteristics and directions. In: WISE, pp. 3–12. IEEE Computer Society, Los Alamitos (2003)
2. Bitsaki, M., Danylevych, O., van den Heuvel, W.J., Koutras, G., Leymann, F., Mancioppi, M., Nikolaou, C., Papazoglou, M.P.: An architecture for managing the lifecycle of business goals for partners in a service network. In: Mähönen, P., Pohl, K., Priol, T. (eds.) ServiceWave. LNCS, vol. 5377, pp. 196–207. Springer, Heidelberg (2008)
3. Caswell, N.S., Nikolaou, C.N., Sairamesh, J., Bitsaki, M., Koutras, G.D., Iacovidis, G.: Estimating value in service systems: a case study of a repair service system. IBM System Journal 47(1), 87–100 (2008)
4. McGovernan, D.: An introduction to BPM and BPMS. Business Integration Journal (April 2004)
5. Papazoglou, M.P., van den Heuvel, W.J.: Service oriented architectures: approaches, technologies and research issues. VLDB J. 16(3), 389–415 (2007)
6. Havey, M.: Essential Process Modeling. O'Really (2005)
7. Object Management Group (OMG): MDA Guide Version 1.0.1, Document Nr: omg/2003-06-01 (June 2003)
8. Sendall, S., Kozaczynski, W.: Model transformation: The heart and soul of modeldriven software development. IEEE Software 20(5), 42–45 (2003)
9. Ouyang, C., Dumas, M., ter Hofstede, A.H.M., van der Aalst, W.M.P.: From BPMN process models to BPEL web services. In: ICWS, pp. 285–292. IEEE Computer Society, Los Alamitos (2006)
10. Hornung, T., Koschmider, A., Mendling, J.: Integration of heterogeneous BPM schemas: The case of XPDL and BPEL. In: Boudjlida, N., Cheng, D., Guelfi, N. (eds.) CAiSE Forum. CEUR Workshop Proceedings, vol. 231 (2006) CEUR-WS.org.
11. Decker, G., Kopp, O., Leymann, F., Weske, M.: BPEL4Chor: Extending BPEL for modeling choreographies. In: ICWS, pp. 296–303. IEEE Computer Society, Los Alamitos (2007)
12. Chapell, D.A.: Enteprise Service Bus. O'Really (2004)
13. TeleManagement Forum: Enhanced Telecom Operations Map The Business Process Framework For The Information and Communications Services Industry, http://www.tmforum.org
14. Object Management Group (OMG): Business Process Modeling Notation Specification (February 2006)
15. RosettaNet Consortium: Partner Interaction Processes (PIPs), http://www.rosettanet.org

Building Scientific Workflow with Taverna and BPEL: A Comparative Study in caGrid

Wei Tan[1], Paolo Missier[2], Ravi Madduri[3], and Ian Foster[1]

[1] Computation Institute, University of Chicago and Argonne National Laboratory, Chicago, IL, USA
[2] School of Computer Science, University of Manchester, Manchester, UK
[3] Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL
wtan@mcs.anl.gov, pmissier@cs.man.ac.uk, madduri@mcs.anl.gov, foster@mcs.anl.gov

Abstract. With the emergence of "service oriented science," the need arises to orchestrate various services to facilitate scientific investigation -- that is, to create "science workflows." In this paper we summarize our findings in providing a workflow solution for the caGrid service-based grid infrastructure. We choose BPEL and Taverna as candidate solutions, and compare their usability in the full lifecycle of a scientific workflow, including service discovery, service composition, workflow execution, and workflow result analysis. We determine that BPEL offers a comprehensive set of primitives for modeling processes of all flavors, while Taverna provides a more compact set of primitives and a functional programming model that eases data flow modeling. We hope that our analysis not only helps researchers choose a tool that meets their needs, but also provides some insight on how a workflow language and tool can fulfill the requirement of scientists.

1 Introduction

More and more data and computation resources used by the scientific community are built on a service-oriented architecture (SOA) [1]. Given the proliferation of web services, service-oriented science [2] is becoming an emerging paradigm in facilitating scientific investigation, and scientific workflow has become an important approach to orchestrate various services [3]. For example, caGrid [4] is the service-based grid software infrastructure that underpins the cancer Biomedical Informatics Grid. This infrastructure, based on the Globus Toolkit [5], enables the sharing of information and analytical resources (via grid services). By this means it helps domain scientists to easily contribute to and leverage caBIG resources, accelerating biomedical research in a multi-institutional environment.

There are already many languages, tools and systems exist for scientific workflow [6]. Through a comprehensive survey on existing workflow tools [7], the caGrid team decided to choose Taverna [8] and BPEL [9] as candidate workflow solutions: as Taverna is representative of many scientific workflow systems, while BPEL is an well-accepted standard in business domain and is gaining momentum in science.

G. Feuerlicht and W. Lamersdorf (Eds.): ICSOC 2008, LNCS 5472, pp. 118–129, 2009.

BPEL: WS-BPEL (Web Service-Business Process Execution Language, or BPEL for short) is a meta-model and an XML-based specification for describing the behavior of a business process that is composed of Web services and also exposed as a Web service. Although originally designed for business workflows, BPEL has also attracted attention from the scientific community because of its support for the SOA paradigm. BPEL can be seen as a good representative of those languages originated from business domain and are now been adopted by the scientific community.

Taverna: Developed in the UK by the myGrid consortium (http://www.mygrid.org.uk), Taverna is an open-source workbench for the design and execution of scientific workflows. Aimed primarily at the life sciences community, its main goal is to make the design and execution of workflows accessible to bioinformaticians who are not necessarily experts in web services and programming. A Taverna workflow is a linked graph of *processors*, which represent Web services or other executable components, each of which transforms a set of data inputs into a set of data outputs. These workflows are represented in the Scufl language (using an XML syntax), and executed according to a functional programming model [10]. The data-driven model is briefly presented in Section 4. Taverna also provides a plug-in architecture so that additional applications, such as secure Web Services, can be populated to it. The caGrid plug-in recently implemented by members of our group [11] is an example.

The design and implementation of workflow systems for scientific purposes has been a subject of considerable research [6]. The goals of this paper are to communicate practical experiences based on our work in the caGrid project. In this analysis, we consider the entire scientific workflow lifecycle, from service discovery to service composition, workflow execution, and workflow result analysis. The analysis is based on our understanding of caGrid's requirements for a workflow language and tooling, but we believe is also applicable to other areas in data intensive and exploratory science. We hope that our work not only helps researchers choose a tool that meets their needs, but also provides some insight on how a workflow language and tool can fulfill the requirement of scientists.

In our comparison of BPEL and Taverna we consider not only the two workflow languages but also their associated tooling. This is because scientific workflow users are generally scientists that have expertise in their specific domain (biology, physics, astronomy, etc.) but understandably limited knowledge of IT technology and thus require easy-to-use tooling. In the remaining of this paper, the term Taverna is used to refer to both the Scufl workflow language that Taverna uses and the Taverna tool, while BPEL to both the language and the open source tools supporting it.

In the remainder of this paper we first present a caGrid use case and then examine the lifecycle and features of scientific workflows. Then, we compare Taverna and BPEL from three perspectives: service discovery, service composition and workflow execution, and workflow results analysis. Finally, we draw conclusions.

2 A caGrid Use Case

We present a caGrid use case that relates to the querying of semantic data in cancer research. The use of a standardized metamodel and semantic annotation to enable the formal description and harmonized use of data is a primary feature that caGrid moves beyond the basic grid infrastructure.

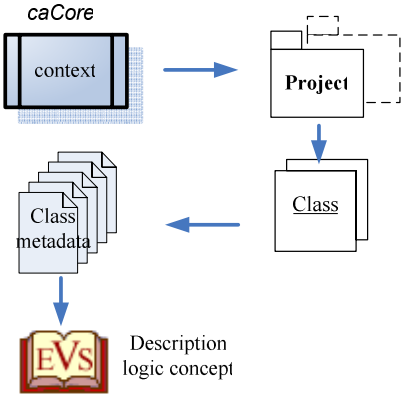


Fig. 1. The caGrid use case used in the paper

In the use case illustrated in Fig. 1, a user wants to query description logic concepts that relate to a particular context, namely "caCore." First, the user queries all projects related to context "caCore"; second, they find UML classes in each project; third, they use project and UML class information to query the semantic metadata; and finally, they retrieve the concept code.

3 The Lifecycle and Features of Scientific Workflows

To define the scope of the comparative study, we first discuss the lifecycle of a scientific workflow. This lifecycle involves four stages: discover relevant data/analytical services, compose these services into a workflow, execute workflow, and analyze the results (see Fig. 2).

1. *Discover relevant data/analytical services*. Data/analytical services are developed, owned and maintained by different institutions, organizations, etc. Usually the URL of these services is not well-known. Moreover, the scientific community is too autonomous to share a common terminology, so domain knowledge is needed to get the exact semantics of the services whose syntax is already known.
2. *Compose these services into a workflow*. After the individual services are found, the next step is to compose them into a workflow. This step involves the addition of data and control dependencies between services; it may also involve data transformations between services' invocation.
3. *Execute workflow*. A workflow definition is sent to an engine for execution. The engine invokes the services in the pre-defined order.
4. *Analyze results*. Scientific workflow is for the purpose of exploratory research, and therefore, the intermediate results generated by component services, as well as the final results yield by the workflow, are of great value and deserve to be analyzed carefully. Scientific researches are usually undertaken in an iterative manner so the analysis results often initiate another round of workflow modeling/execution.

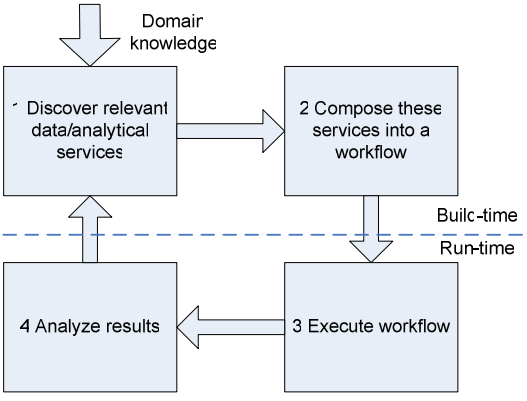


Fig. 2. Lifecycle of a scientific workflow

We summarize some features of scientific workflows in caGrid, and these features can also be seen as challenges encountered when providing a scientific workflow solution in a more general sense (the number of the bullets represents their position in the lifecycle shown in Fig. 2).

- (1) Resources are highly distributed. Compared to business domain, users in scientific domain usually use services owned by other organizations, like data storage, high-performance computing, etc.
- (2) Data-flow oriented. Data is considered to be the first-class citizen in scientific workflows, because scientific workflows are mostly pipelines of parallel data processing. In a data flow, tasks and links represent data processing and data transport, respectively; parallel execution of independent tasks is desired to be modeled for free -- tasks can execute once their input are ready.
- (2&3) Large scale. Scientific workflows often contain many tasks, involve large data sets, and require intensive computation. The modeling tool should make it easy to model such complex workflows.
- (4) Data analysis and provenance is an important step and the workflow execution can be in an iterative manner.

In the rest of this paper we highlight some of the differences between the BPEL and Taverna, from the point of view of their impact on the users' experience in the lifecycle of scientific workflows. The discussion is organized according to the lifecycle model of Fig. 2.

4 Support for Service Discovery

As suggested in Fig. 2, a user's first task involves finding appropriate services that can be composed into a workflow. In a Grid setting, these services are virtualizations of data storage, computation capability or other resources. Service endpoints are not naturally known to users, either because users are not familiar with the service itself, or because the service deployment may have changed in time. Support for service discovery is therefore needed.

Taverna offers two levels of support for this. Firstly, it is often the case that a website is known to host one or more services. In these cases, a *scavenger* meta-service can be used to locate endpoints within the site which correspond to valid WSDL service definitions. The WSDL is automatically analyzed and a description of the service is added to the Taverna workbench's library, ready to be used in workflows.

The Taverna plug-in framework simplifies the creation of new scavengers that may offer advanced service discovery features. As part of the caGrid project, for example, we have developed a *caGrid scavenger* that supports semantic/metadata based query to caGrid services. Users can use multiple query criteria to get the list of desired services. For example, they can query the services that are developed by *Ohio State University*, whose names are *CaDSRService*, and with the class *Project* as output. Through this query we find the matching service for the first step in the use case shown in Fig. 2 – use context to get related projects.

As a second, more general-purpose option, a semantic discovery facility called *Feta* [12] also offered natively as part of the Taverna distribution. Feta includes a semantic service registry that maintains annotated description of services, and can be searched using terms from a publicly available ontology. The annotations describe (1) the task performed by a service, for example *bioinformatics task*, (2) the type of resource used by the service, e.g. *bioinformatics data resource*, (3) the types of input data it accepts and of output data it produces (*protein structure*, for example), and more. The terms used in the annotations belong to the myGrid ontology [13], a controlled ontology of terms for the bioinformatics domain.

These discovery facilities stand in contrast with the lack of analogous integrated tools for BPEL design environments. To the best of our knowledge, no open-source BPEL tool is available that works with a service query component in an integrated way.

5 Service Composition and Workflow Execution

The second and third phase of the lifecycle involves composing the discovered services into complete workflows and executing them. In this section we focus on the modeling style, the definition of data, the iteration strategies adopted by BPEL and Taverna, respectively, and their influence to the run-time engine.

5.1 Data-Driven vs. Control-Driven Modeling

When modeling a workflow, users are confronted with the choice among the different modeling paradigms offered by Taverna and BPEL. While the former follows a pure data flow approach to workflow modeling and execution, the latter exposes a fundamentally procedural language.

In a data flow model, the workflow is described as a graph where nodes represent processors that can be executed on input provided along the incoming arcs, and whose output is forwarded to other processors through outgoing arcs. In this model, the order in which the processors are executed is determined primarily by the order in which the data appears on the various inputs. Any processor for which the input data is available can be scheduled for execution.

In Taverna, scheduling is simple: processors are executed as soon as possible, in a greedy fashion as long as a new execution thread can be started (a limit on the number of threads can be defined on the scheduler). This means, in particular, that parallelization of processor execution is managed by the scheduler, based on the available data, without the need for explicit user directives. Also, the order of execution of two processors that have no data dependencies amongst each other may be different for different executions of the same workflow, even on the same input, due to the possible variations in execution speeds of some of the other processors.

In contrast, a procedural workflow language like BPEL includes the explicit definition of the control flow that determines the order of execution of the processors. In particular, parallel execution of independent processors must be specified explicitly.

A comprehensive analysis on the differences and relative merits of control-driven and data-driven execution is beyond the scope of this paper. A more in-depth discussion can be found in [14]. In the rest of this section we focus on the specific differences between Taverna and BPEL, summarized in Table 1.

Table 1. Comparison of BPEL and Taverna (Scufl) w.r.t. control/data-flow

	BPEL	Taverna (Scufl)
Activities in model	Basic and structure activities	Processors as data processing units with in/output ports
Semantics of links	Transfer of control	Transfer of data
Data definition	Explicitly defined (global variables)	Implicit defined (processor's input/output)
Data initialization	Complex data type need to be explicitly initialized	Automatically
Control logic	Full-fledged: sequence, conditional, parallel, event-triggered, etc	Limited: sequential, parallel and conditional
Parallel execution	Defined in <flow> or <ForEach>	By default

5.2 Implicit vs. Explicit Definition of Data

Complementary to the control model described in the previous section is the data specification model. In Taverna, processors have input and output *ports* with an associated data type, and data travels from the output port of a processor to the input for of one or more downstream processors. No other data structure specification is needed besides the port types, and interaction among processors is defined entirely by the arcs in the dataflow graph.

In contrast, BPEL requires the explicit definition of variables to hold data structures that are meant to be shared amongst activities; furthermore, each activity can be specified as either a producer or a consumer for values associated to a variable.

Although BPEL's requirement for explicit data definition takes additional effort, it also brings about flexibility. For example, in BPEL you can easily define a data that controls the overall flow but is not the input/output of any activities, but in Taverna you have to add a processor to hold this data (as either input or output).

In BPEL, variables of complex type, must also be initialized prior to their first use (i.e., by means of the <*copy*>syntactic construct – see Section 8.4.2 of the WS-BPEL Specification in [9]). In contrast, Taverna provides a special built-in processor, called an *XML splitter*, which automatically pulls apart a complex XML message defined in a WSDL interface so that its components can be easily accessed by other user-defined processors. An example of its use is provided in the next sub-section.

5.3 Implicit vs. Explicit Iteration on Data

Each port in a Taverna processor has a type, which is either a simple type value (i.e., a string, a number) or a list, possibly nested, of simple type values. As part of normal processing, it may be the case that an input port receives a value of a type that does not correspond exactly to its declared type. A processor that outputs a value of type "list of strings," for example, can legally be connected to a processor with an input port of type "string." Taverna interprets this type mismatch as an indication that the destination processor must be invoked repeatedly, once for each element of the input list. This behavior is consistent with Taverna's functional programming model, whereby the application of a function *f* with a formal argument of type *t*, to an actual parameter *x* of type list(*t*), is interpreted as (map *f x*).

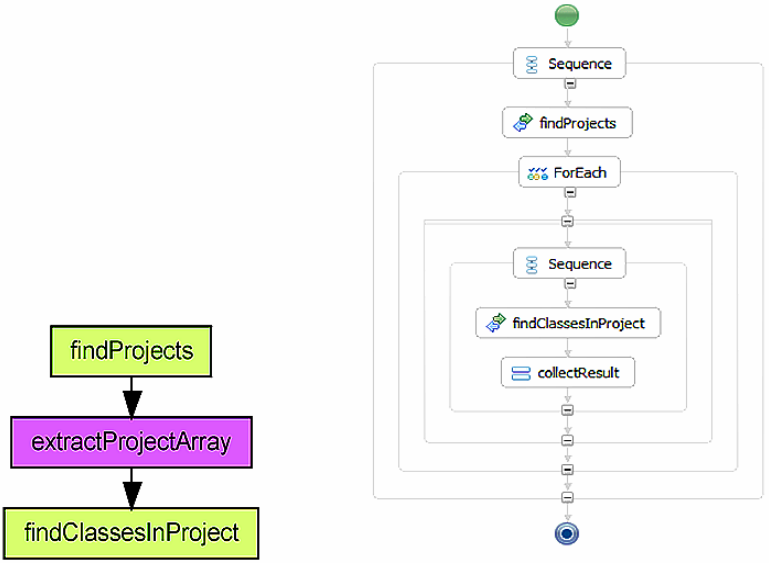


Fig. 3. Implicit iteration

In general, when mismatches appear simultaneously on multiple input ports, Taverna performs either a cross-product (i.e., a Cartesian product) or a dot product (if the cardinalities of the two lists are the same) involving the elements of each of the unexpected lists. Users may explicitly choose which of these two iteration strategies is appropriate for each processor. The implicit iteration feature is commonly used in Taverna scientific workflows. The implication, from the user's perspective, is that the design of a service can be simplified by assuming that it will manage individual data items, while the execution engine takes care of managing input collections.

In Fig. 3, see an example from the caDSR (Cancer Data Standards Repository) service that access and generate the information related to caGrid standard metadata. caDSR has two operations: *findProjects* and *findClassesInProject*. Operation *findProjects* returns a set of projects (i.e., an array *Project []*); *findClassesInProject* receives an instance of data type *Project* and find all the UML classes in this project. Fig. 3 illustrates the Taverna and BPEL presentation of how to connect them into a workflow. The left and right parts are Taverna and BPEL representation, respectively. In the left part, the output of *findProjects* is put to an xml-splitter which extracts out the project array, and sends it to *findClassesInProject*. In the right part, since BPEL does not have an implicit iteration mechanism, a <*ForEach*> construct is added and configured to iterate on project array. After each invocation of *findClassesInProject*, result data need to be collected and merged into the final results set.

From this example one can see that, BPEL handles the iteration like an imperative programming language, a <*ForEach*> construct and the iteration method (a counter, an array or an expression) is to be configured. It is verbose and exposes too many implementation details to the end users (and thus error-prone). Taverna deals with this issue in a straightforward way -- its implicit iteration framework requires (in the simplest cases) no additional configuration, and the user simply connects an output containing a collection of items into an input that consumes a single item of the same type. This leaves the complexity to the workflow engine instead of the users.

Again, as an imperative language, BPEL offers more flexibility in handling advanced iteration strategies. For an example, BPEL can handle this issue: an activity receives two lists of inputs, needs a special kind of dot-product iteration over them, with a special “correlation” mechanism (like, classes and projects with the same developer should be combined.)

For space limitation, in Fig. 4 we only show the completed Taverna workflow for the caGrid use case in Fig. 1. There are four caGrid processors (*findProject*, *findClassesInProject*, *findSemanticMetadataForClass*, and *searchDescLogicConcept*) that represent caGrid services, and more “shim” processors for data transformation between caGrid processors.

6 Workflow Result Analysis

The final phase of the workflow lifecycle, namely analysis of the results, is increasingly perceived as of great importance within the e-science community [15]. The *provenance* of a piece of data produced by an arbitrary process is a complete account of how that piece of data was computed, starting from user input and taking into account intermediate results produced by the processors involved in the computation. Business and scientific workflows may differ in both their requirements and their ability to track data provenance, in particular with regards to the *precision* of provenance information.

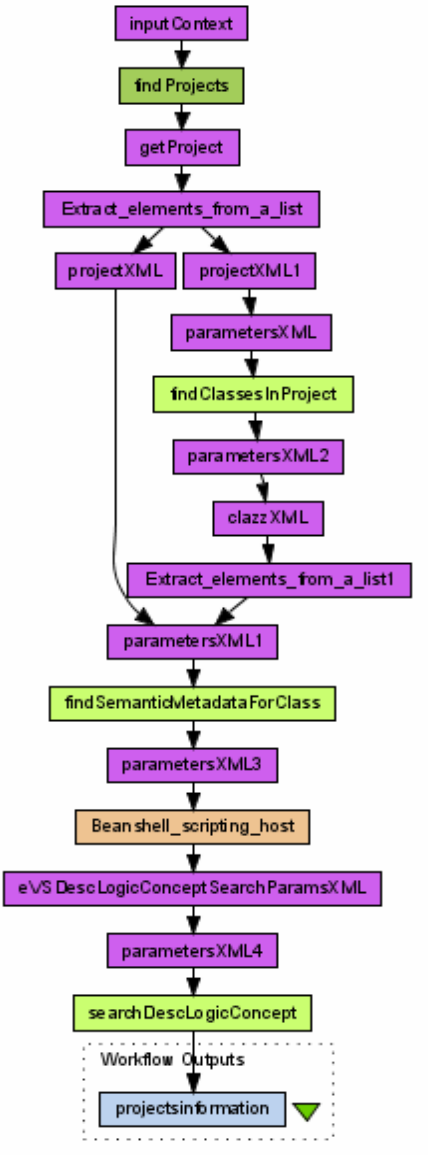


Fig. 4. Completed Taverna workflow for the caGrid use case in Fig. 1

Precision, in this case, denotes the levels of detail at which provenance can be traced, and depends on the unit of information that the workflow engine can observe during execution. When dealing with Web Services, both in BPEL and Taverna, the atomic unit of information that flows through a processor is an XML document, for instance “purchase order” for a business process, or an XML-formatted description of a protein in the case of a scientific process. The black-box nature of the Web Services

that produce and consume these documents limits the ability to track its individual elements. For instance, consider a service that takes a purchase requisition request document as input, and returns a purchase order document. While it is likely that specific elements within the purchase order depend on only some of the input document elements, this fine-grained dependency is hidden within the service logic: from the point of view of provenance, the service is a black box, because the nature of the data transformation they implement is not exposed through the WSDL interface. Thus, the only data dependency that can be safely used in provenance tracking is that the entire purchase order depends on the entire purchase requisition request. The black-box nature of the service limits the degree of precision with which provenance of the output can be tracked: the granularity of traceable provenance is that of entire XML documents, rather than that of their composing elements.

As we mentioned, this problem affects both BPEL and Taverna. Unlike BPEL, however, Taverna is not limited to using processors that are implemented as Web Services; processors types include local Java classes, as well as *beanshells*, or small interpreted Java programs. This makes it quite natural for Taverna workflows to handle simple types, such as strings, as well as collections of elements of these types that often represent sets of scientific data products. In this case it is important to be able to track the provenance of each of these products individually. Our caGrid use case, for example, involves a one-to-many association between Projects and their UML classes, which is then used to retrieve semantic concepts associated to project classes. For provenance information to be useful, here we cannot simply state that "the collection of the concepts depends on the collection of input projects," because this is as trivially true as it is uninteresting. Instead, we must be able to determine that the presence of a specific concept in the output is due to a specific project being present in the original input.

An important example of this fine-grained data manipulation is the "packing" and "unpacking" of complex XML data, something that can be achieved automatically using XML splitters, as mentioned in Sec.5.2 and 5.3. In some cases, this may enable provenance tracking through the internal element of XML documents, for instance it may be possible to trace the *originator* element of a purchase order back to some specific workflow input, at a stage in the process prior to its use as part of the order.

In our preliminary experiments on provenance tracking in Taverna, performed within the myGrid team, we have been able to achieve high precision in many practical cases, namely when simple values are composed into collections or into complex XML messages in a way that is visible to the engine, i.e., by means of dedicated packing and unpacking processors.

As a corollary to this investigation, we have also been arguing that processors that map entire collections to new collections (i.e., without any iteration being exposed to the workflow engine) should be annotated, where possible, with an indication of properties of the mapping that help provenance tracking. A detailed discussion of the promises and limitations of this idea can be found in [16].

Other approaches to tracking provenance through Web Services involve the explicit semantic annotation of the involved services. This *semantic provenance overlays* approach is really complementary to the problem discussed in this section, and early experiment done on Taverna show promising results [17].

7 Conclusion and Future Work

From our experience in using both Taverna and BPEL as the candidate solutions for caGrid workflow, we have the following conclusions:

1. Taverna provides a compact set of primitives that eases the modeling of data flow. This functional-programming manner allows users to tell "what to do" instead of "how to achieve it."
2. BPEL offers a comprehensive set of primitives to model processes of all flavors (control-flow oriented, data-flow oriented, event driven, etc), with full feature (process logic, data manipulation, event and message processing, fault handling, etc). BPEL is also flexible enough to handle complex processing logic, although a little bit verbose in modeling basic data flow.
3. As a tool-suite, Taverna provides better support in the whole lifecycle of scientific workflows, including service discovery and results analysis, than the existing open-source BPEL tools do.

We do not mean to indicate that Taverna is better than BPEL, or vice versa. We would rather say that Taverna better fits the requirement of modeling a data flow, and the open source community has provided a handy workbench that consists of the modeling and the execution tools. We also acknowledge nice features of BPEL engines. For example, BPEL engines typically run inside application servers and are with persistent state storage, which offer more reliability and scalability. This is important for those long-running and computation-intensive workflows. For now the Taverna engine does not provide these capabilities.

At the same time, we suggest a promising *multi-stage modeling approach* in adapting BPEL to scientific workflow, leveraging its capability and retaining the simplicity. That is, the scientists use a model which is intuitive to them, and transform this model into a standard BPEL model automatically, through a macro-expansion procedure. This BPEL model can be orchestrated by a BPEL-compliant engine. Actually this approach has already been adopted by existing research efforts [18]. In future, we also plan to investigate the possibility to provide a BPEL-centric tool set where discovery and result analysis tools are included.

Acknowledgement

We thank Taverna team, especially Mr. Stian Soiland-Reyes at University of Manchester, for the great help in using Taverna and developing plug-ins for it. We also thank the constructive comments from anonymous reviewers. This project has been funded in part with Federal funds from the National Cancer Institute, National Institutes of Health, under Contract No. N01-CO-12400.

References

1. Krishnan, S., Bhatia, K.: SOAs for Scientific Applications: Experiences and Challenges. In: Proc. IEEE International Conference on e-Science and Grid Computing (2007)
2. Foster, I.: Service-Oriented Science. Science 308(5723), 814–817 (2005)

3. Tan, W., et al.: Workflow in a Service Oriented Cyberinfrastructure Environment. In: Cao, J. (ed.) Cyberinfrastructure Technologies and Applications. Nova Science Publishers (2008)
4. Saltz, J., et al.: caGrid: design and implementation of the core architecture of the cancer biomedical informatics grid. Bioinformatics 22(15), 1910–1916 (2006)
5. Foster, I.: Globus Toolkit Version 4: Software for Service-Oriented Systems. Journal of Computer Science and Technology, 2006 21(4), 513–520 (2006)
6. Taylor, I.J., et al.: Workflows for e-Science: Scientific Workflows for Grids. Springer, Heidelberg (2007)
7. ICR Workflow Working Group, Tool Reviews (2007), http://gforge.nci.nih.gov/docman/view.php/332/7509/icr_workflow_tool_review_2007.doc
8. Oinn, T., et al.: Taverna/myGrid: aligning a workflow system with the life sciences community. In: Taylor, I.J., et al. (eds.) Workflows for E-science: Scientific Workflows for Grids, pp. 300–319. Springer, Guildford (2007)
9. OASIS, Web Services Business Process Execution Language Version 2.0 (2007), http://docs.oasis-open.org/wsbpel/2.0/CS01/wsbpel-v2.0-CS01.html
10. Turi, D., et al.: Taverna Workflows: Syntax and Semantics. In: Proc. 3rd e-Science Conference, Bangalore, India (2007)
11. Tan, W., et al.: Orchestrating caGrid Services in Taverna. In: Proc. IEEE International Conference on Web Services (ICWS 2008), Beijing, China (2008)
12. Lord, P., et al.: Feta: A light-weight architecture for user oriented semantic service discovery. In: Gómez-Pérez, A., Euzenat, J. (eds.) ESWC 2005. LNCS, vol. 3532, pp. 17–31. Springer, Heidelberg (2005)
13. Wolstencroft, K., et al.: The myGrid ontology: bioinformatics service discovery. International Journal of Bioinformatics Resesearch and Applications 3(3), 303–325 (2007)
14. Shields, M.: Control- Versus Data-Driven Workflows in Workflows for E-science: Scientific Workflows for Grids. In: Taylor, I.J., et al. (eds.), pp. 167–173. Springer, Heidelberg (2007)
15. Simmhan, Y., Plale, B., Gannon, D.: A survey of data provenance in e-science. SIGMOD Record 34(3), 31–36 (2005)
16. Missier, P., et al.: Data lineage model for Taverna workflows with lightweight annotation requirements. In: Proc. Second International Provenance and Annotation Workshop, University of Utah, Salt Lake City, Utah (2008)
17. Zhao, J., et al.: Using semantic web technologies for representing E-science provenance. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 92–106. Springer, Heidelberg (2004)
18. Tan, W., Fong, L., Bobroff, N.: BPEL4Job: A fault-handling design for job flow management. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749, pp. 27–42. Springer, Heidelberg (2007)

Second International Workshop on Web APIs and Services Mashups (Mashups 2008)

Introduction: Second International Workshop on Web APIs and Services Mashups (Mashups 2008)

Cesare Pautasso[1], Stefan Tai[2], and E. Michael Maximilien[3]

[1] Faculty of Informatics
University of Lugano, Switzerland
cesare.pautasso@unisi.ch
[2] Karlsruhe Service Research Institute (KSRI)
Karlsruhe Institute of Technology (KIT), Germany
stefan.tai@kit.edu
[3] IBM Almaden Research Center
San Jose, CA, USA
maxim@us.ibm.com
http://icsoc-mashups.org

1 Overview

The Mashups workshop series is about the convergence of Service computing and the Web 2.0 in the form of mashups: end-user-oriented compositions of Web-accessible services, APIs, and data sources, sometimes taking advantages of social and collaborative nature of Web 2.0 content and services. The interaction and integration of Service computing and Web 2.0 expose various issues, challenges and problems that have to be addressed in order to make the vision of a fully programmable Web become a reality. In this space, a flurry of new languages, tools, platforms, and solutions for end-user service reuse, assembly and composition is displacing current legacy distributed system applications and middleware. The Mashups workshop series is the international forum intended to foster a conversation within the service-oriented computing community around the research, emerging technologies and novel applications that appear as the Web 2.0 paradigm, as well as the more established Web services middleware technology. As these two communities continue to merge, the Mashups workshop is a venue that can help bring about the innovation potentials of the programmable Web and accelerate their applications and disseminations.

The following papers constitute the proceedings of Mashups'08 the second international workshop in a successful series of workshops that look specifically at the engineering, scientific, social, and business concerns and challenges encountered in the area of Web mashups.

Mashups'08 was held on December 1st, 2008, co-located with International Conference of Service-Oriented Computing (ICSOC) in Sydney, Australia. The workshops call for papers targeted submissions on topics spanning:

– Programming models (languages, frameworks, and platforms) for the composition of Web-accessible services, APIs, content, and data of all kinds and of

G. Feuerlicht and W. Lamersdorf (Eds.): ICSOC 2008, LNCS 5472, pp. 133–135, 2009.

different architectural styles (e.g., SOAP, REST, RSS, and Atom Publishing Protocol)
- Quality of mashups, including reliability and security, and management approaches
- Understanding social and economic factors in the creation, the acceptance, and the sustainability of services mashups, including software-as-services (SaaS) markets and services marketplaces, digital communities, as well as directory, pricing, and contracting models

The full-day workshop program featured two invited keynote speakers, and five research paper presentations. The topics of the presentations are representative of the current state of research in the area. They cover:

- the demonstration of visual mashup development tools for non-programmers,
- the discussion of social aspects of mashup applications,
- the analysis of the mashup ecosystem,
- the observation of the impact of mashup technology to IT organizations,
- the usage of mashup visual tools to awake the interest of young inexperienced end-users in computer science,
- the performance optimization required to scale mashup run-times, and
- the innovative mashup application of cloud storage services.

The organizers would like to thank Nick Hodge (Microsoft, Inc.) and Pamela Fox (Google, Inc.) for their inspiring keynote presentations, the papers authors, the panelists, the participants, and the international program committee for their help in also making this years edition of the workshop a success. We are grateful to Microsoft for their financial support and to the ICSOC conference organizers for supporting Mashups'08.

2 Organization

Program Chairs

E. Michael Maximilien, IBM Almaden, USA
Cesare Pautasso, University of Lugano, Switzerland
Stefan Tai, Karlsruhe University, Germany

Publicity Chair

Nirmit Desai, NC State University

Program Committee

Gustavo Alonso, ETH Zurich, Switzerland
Mehmet Altinel, Anvato, Mountain View, CA
Brian Blake, Georgetown University
Christoph Bussler, MercedSystems, Inc, USA
Schahram Dustdar, Vienna University of Technology
George Feuerlicht, University of Technology, Sydney
Robert Ennals, Intel Research, Berkeley, CA
Gregor Hohpe, Google, Inc.
Christine Legner, European Business School, Germany
Mehdi Jazayeri, University of Lugano, Switzerland
Anant Jinghran, IBM Silicon Valley Labs
Rania Khalaf, IBM T. J. Watson Research Center
Jonathan Marsh, WSO2
Ravi Nemana, Services Science at UC Berkeley
Duane Nickull, Adobe Systems
Dave Nielsen, Independent Consultant
Ajith Ranabahu, Wright State University and Apache Software Foundation
Amit Sheth, Kn.o.esis Center, Wright State University
Ashutosh Singh, IBM Almaden Research Center
Kunal Verma, Accenture Research Labs

Innovation in the Programmable Web: Characterizing the Mashup Ecosystem

Shuli Yu and C. Jason Woodard

School of Information Systems
Singapore Management University
{shuli.yu.2004,jwoodard}@smu.edu.sg

Abstract. This paper investigates the structure and dynamics of the Web 2.0 software ecosystem by analyzing empirical data on web service APIs and mashups. Using network analysis tools to visualize the growth of the ecosystem from December 2005 to 2007, we find that the APIs are organized into three tiers, and that mashups are often formed by combining APIs across tiers. Plotting the cumulative distribution of mashups to APIs reveals a power-law relationship, although the tail is short compared to previously reported distributions of book and movie sales. While this finding highlights the dominant role played by the most popular APIs in the mashup ecosystem, additional evidence reveals the importance of less popular APIs in weaving the ecosystem's rich network structure.

Keywords: API, mashup, social network, power law, long tail, small world.

1 Introduction

The emergence of a new generation of web-based technologies, collectively dubbed "Web 2.0" [1], has fueled the growth of applications such as wikis and blogs that make it easier for users to publish their own content. A subset of these technologies have set the stage for an even more profound transformation by enabling users to go beyond static publishing and create their own web applications using powerful building blocks provided by third parties. This paper explores the recent explosion of these personalized applications, called *mashups*, and the application programming interfaces (APIs) they build upon.

The term "mashup" is borrowed from pop music, where it denotes remixing songs (or parts of songs) to create new derivative works. Similarly, web-based mashups are created by integrating data from one or more sources to create a new application, typically in a way that hides the details of the source applications to provide a seamless experience for the user [2]. Major companies like Google, Amazon and eBay have provided interfaces to many of their services at little or no cost, allowing individuals and other businesses to create composite applications with novel functionality. As more firms choose to provide APIs for public use, the number of opportunities to combine these APIs in new ways increases exponentially. Each new mashup may then attract its own base of users, further extending the market reach of the API providers.

G. Feuerlicht and W. Lamersdorf (Eds.): ICSOC 2008, LNCS 5472, pp. 136–147, 2009.

Despite the potential importance of this trend, few empirical studies have attempted to characterize the ecosystem of Web 2.0 mashups and APIs in a general way. Most of the existing literature is focused on the concerns of stakeholders in a particular domain (e.g., health librarians [3] or digital journal publishers [4]), the legal and policy issues associated with remixing content [5, 6], or the underlying technologies [7, 8]. A number of classification schemes for mashups have been proposed [1, 9], but we are unaware of any studies that apply these schemes in an empirical setting.

The task of characterizing the mashup ecosystem is made more complicated—and more interesting—by the fact that the web of relationships among mashups and APIs has evolved along with the populations of each. To understand the dynamics of these relationships, we need to investigate the process by which mashup creators choose APIs to build on, which in turn depends on the decisions of API providers (to expose their APIs in the first place, as well as the terms under which they do so) and the expectations of the mashup creators' own target audiences. The importance of these interlinked decisions suggests viewing mashups and APIs as a single evolving network rather than as independent populations of discrete entities. This network-based approach allows us to explore how APIs become connected through mashups, and how these connections influence the overall network structure and the popularity of individual APIs.

This paper presents the results of a preliminary effort to study the API–mashup network using well-established concepts and techniques. Section 2 describes our data set, which was obtained from publicly available sources. We examined the following characteristics of the network, and report the results in sections 3–5 respectively:

- *Graphical network structure.* Visual snapshots of the network illustrate its rapid growth and reveal qualitative structural patterns, most notably a partition of APIs into three tiers, with Google Maps at the center. APIs in Tier 1 and 2 tend to serve as platforms, while those in Tier 2 and 3 often serve as data sources.
- *Degree distributions*. Plotting the cumulative distribution of mashups to APIs reveals a power-law relationship, which is commonly generated by processes in which popular network nodes attract new links at a higher rate than less popular ones [10]. We also assess the extent to which the mashup ecosystem exhibits the "long-tail" property found in studies of books, movies, music and other information goods [11]. Perhaps surprisingly, the distribution of API popularity has a relatively short tail compared to other types of goods.
- *Social network statistics*. Analysis of the API affiliation network provides additional information on the evolving network topology. We find that the links between APIs exhibit the properties of a small-world network [12], suggesting a high level of novelty in mashup designs. While the most popular APIs are responsible for the vast majority of links, the small-world structure is due mainly to the less popular APIs.

Section 6 comments on the implications of our findings for stakeholders in the mashup ecosystem, and concludes with a call for further research.

2 Data: The Programmable Web

To construct a network view of the mashup ecosystem, we turned to the largest online repository of information about Web 2.0 mashups and APIs, ProgrammableWeb.com. This aggregator site provided the most comprehensive listing of mashups and APIs available, including information on which mashups use which APIs.

Our data set consisted of 2664 mashups and 590 APIs that were registered between September 2005 and December 2007. We used this data to create snapshots of the mashup ecosystem at 9 quarterly intervals, beginning in December 2005.

For each snapshot, we created a rectangular matrix with mashups on the rows and APIs on the columns. Each cell in the matrix was assigned a binary value (0 or 1) indicating whether the mashup corresponding to the given row uses the API corresponding to the given column. These relationships can also be represented graphically. In Figure 1a, APIs are shown as boxes and mashups as circles. The line segments connecting them represent instances in which a mashup uses (i.e., builds on) a particular API.

While the API–mashup network is useful for visualizing relationships among APIs and mashups, many network analysis techniques are limited to a single type of entity. For the analysis in Section 5, we therefore followed standard practice in the social network literature [13] and transformed the rectangular (two-mode) API–mashup matrices into square (one-mode) affiliation matrices to study the relationships between APIs. These API affiliations can also be represented graphically (Figure 1b). Two APIs are linked by a line segment if they have been used together in a mashup.

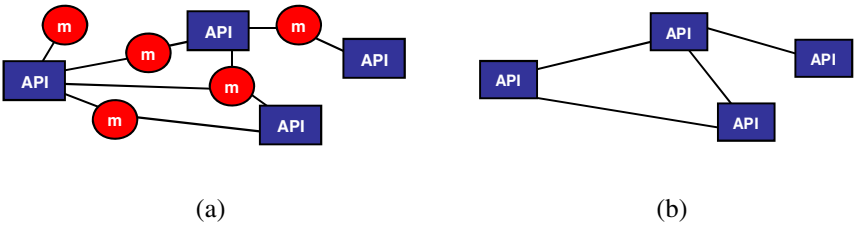


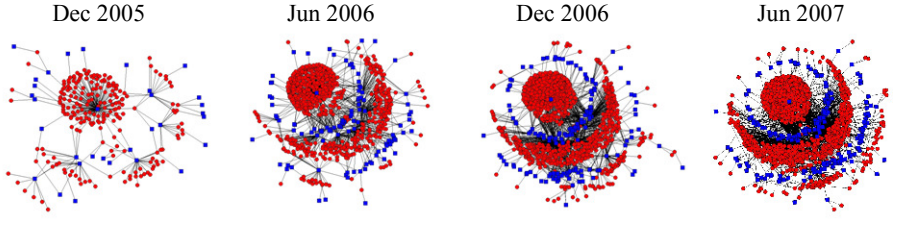
Fig. 1. (a) API–mashup network (b) API affiliation network

3 Graphical Network Structure

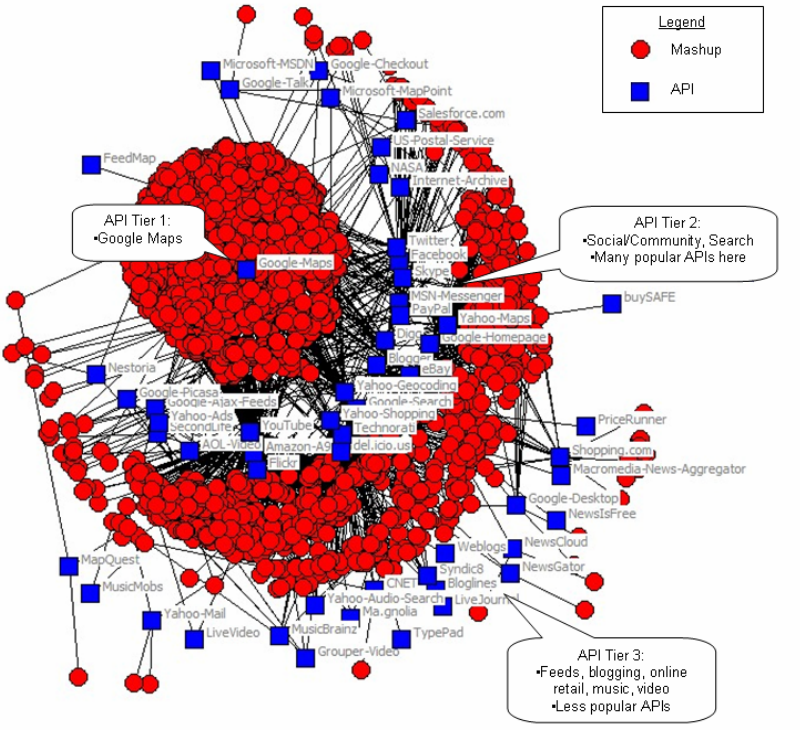
We used NetDraw [14], a popular social network visualization tool, to study the growth of the mashup ecosystem. Figure 2a shows four snapshots of the API–mashup network rendered using NetDraw's node repulsion algorithm. The population of both APIs and mashups grew roughly linearly over time. API growth tended to be slower and more consistent than mashup growth, with APIs growing at a mean rate of 20.1 per month (SD = 6.2) and mashups at a rate of 93.8 per month (SD = 25.4).

While network visualization is as much an art as a science—and the choice of layout algorithm is to some extent arbitrary—it is striking that each snapshot exhibits a distinctive three-tiered structure, with a layer of mashups between each API tier. These layers appeared in all time periods, even as the total number of APIs and mashups increased tenfold.

The enlarged snapshot from December 2007 (Figure 2b) shows a more detailed view of the key APIs and their corresponding mashups. The network clearly centers around Google Maps (Tier 1), with a ring of popular but less central APIs around it (Tier 2) and a constellation of other APIs (Tier 3) on the periphery. This layered structure suggests that Google Maps and at least some of the Tier 2 APIs play the role of *platforms* in the mashup ecosystem. Google Maps, in particular, adds value to the multitude of other APIs that provide spatial data by providing a powerful and convenient way to display this data in a web application. The fact that it is freely available and uses common protocols and data standards makes it an especially attractive choice for mashup developers.



(a)



(b)

Fig. 2. (a) Evolution of the Web 2.0 mashup ecosystem. (b) The API–mashup network in December 2007.

Although the relationships between Tier 2 and Tier 3 APIs are less clear-cut, we observe similar patterns of complementarity between platform-type services and services that supply raw data. APIs for mapping (Google Maps, MS MapPoint), search (Google Search, Yahoo Search), community (Facebook), payment (PayPal) and telephony (Skype) are often combined with APIs that provide data like images (Flickr), video (YouTube, LiveVideo), product details (eBay, Shopping.com, PriceRunner) and news feeds (Technorati, CNET, LiveJournal). Most of the APIs that provide platform services reside in Tier 2, while most of data providers (except the most popular ones) are in Tier 3.

4 Degree Distributions

Before exploring the network structure of the mashup ecosystem in greater depth, we pause to consider a more aggregate phenomenon, namely, the relative frequency with which APIs are used in mashups. This analysis will shed light on how API "market share" is distributed, and provide clues about how some APIs become vastly more popular than others.

4.1 A Power Law: The Rich Get Richer

Figure 3 plots the cumulative distribution of mashups over APIs at two points in time, December 2005 and December 2007. For each distribution, the vertical coordinate of the top-left data point indicates the number of APIs that were used by a single mashup between September 2005 and the given date. The horizontal coordinate of the bottom-right point indicates the number of mashups that used the single most popular API during the same time period. Plotting the distributions on log-log axes reveals a linear relationship characteristic of a power-law distribution.

In a power-law distribution, small events are extremely common and large events are extremely rare [15]. In the context of the mashup ecosystem, this would mean that a large number of APIs are used by few mashups and a small number of APIs are used by many mashups, compared to a bell-shaped pattern in which more and less popular APIs are distributed symmetrically. Such a phenomenon occurs in many markets that are dominated by a few popular products, for example a bookstore that sells a few blockbuster novels in large quantities along with many obscure texts in small quantities.

The power-law pattern is sometimes expressed by the "Pareto principle," which states that 20% of the causes often yield 80% of the effects. In the mashup ecosystem, the distribution is even more lopsided: by December 2007, the top 20% of APIs (121 out of 590) had captured 95% of the market (2535 mashups out of 2664).

Prior research identifies *preferential attachment* as a mechanism that can give rise to power-law distributions [10]. If preferential attachment were operating in the mashup ecosystem, APIs that became popular early in the ecosystem's development would continue to gain mashups at a higher rate than less popular APIs, reinforcing their popularity. We did not test this hypothesis quantitatively, but it seems like a plausible explanation.

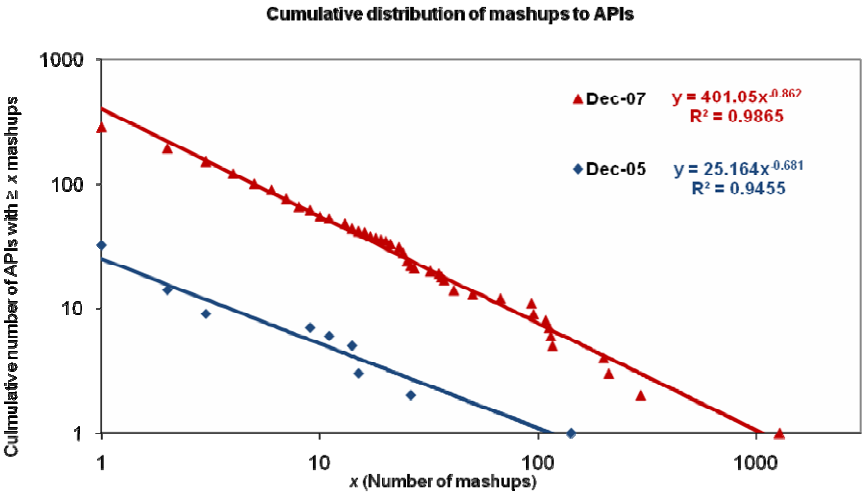


Fig. 3. Cumulative distribution of mashups to APIs for December 2005 and 2007

Although the data for both time periods is consistent with preferential attachment, the strength of the effect appears to have weakened over the period of our study. We computed best-fit lines for the December 2005 and December 2007 distributions, yielding the equations shown in Figure 3. The absolute value of the exponent increased in magnitude from 0.68 to 0.86, which implies a decrease in the fraction of mashups that use the top APIs relative to the fraction of APIs with a small number of mashups. The impact of this change can be seen by looking at the top and bottom ends of the distribution. At the bottom, the number of APIs with only one mashup increased by a factor of 16, from 25 in 2005 to 401 in 2007. At the top, the API with the largest number of mashups (Google Maps) also increased its number of mashups, but only by 9 times, from 114 in 2005 to 1048 in 2007.

4.2 A Long Tail? Yes, But a Short One

The concept of a power-law distribution is frequently associated with the idea of a "long tail." Long-tailed distributions are characterized by a large number of low-frequency occurrences that can cumulatively outweigh the high-frequency ones [11]. Such distributions are common in online retailing, where product selection is not limited by physical storage restrictions or holding costs, and consumers can easily find specific products by searching online or acting on recommendations, resulting in an overall high volume of sales from niche products [16]. Since the mashup ecosystem is similarly unencumbered by physical constraints, it is plausible that the number of APIs with few mashups could be so numerous that they form a long tail and capture the lion's share of the market.

In 2007, Kilkki [17] proposed a mathematical formula to model long-tailed distributions:

$$F(x) = \frac{\beta}{(\frac{N_{50}}{x})^{\alpha} + 1}$$

When products (in this context, APIs) are ranked according to their volume or share, $F(x)$ represents the share of total volume covered by products up to rank x. In this model, three parameters determine the size of the tail: (1) N_{50} is the number of products that cover half of the total volume; (2) α is the factor that defines the form of the function by describing the steepness of slope in the middle part of the function; and (3) β is the total volume of the distribution, including latent demand suppressed by the current structure of the product market. Kilkki applied the model to a wide range of data sets, including book sales and movie viewership in the United States. He found that the distribution of book sales has a much longer tail (N_{50} = 30714, α = 0.49 and β = 1.38) than movie viewership (N_{50} = 56, α = 0.82 and β = 1.60).

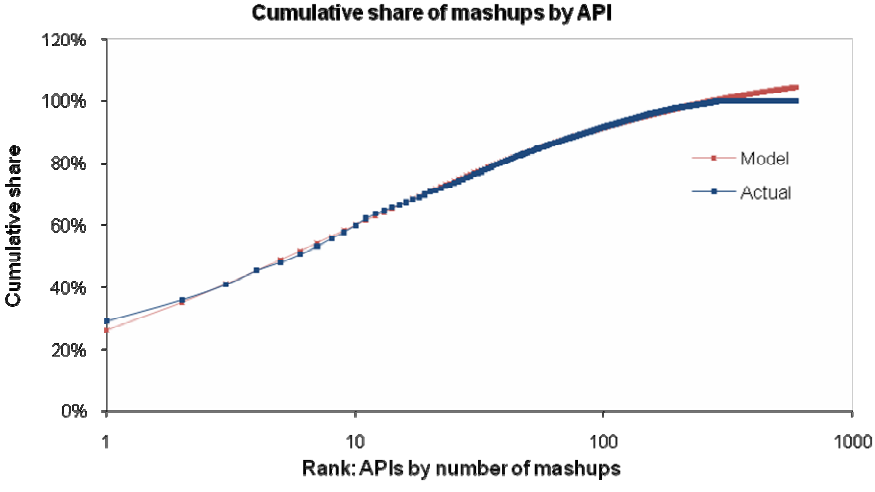


Fig. 4. Cumulative share of mashups by API in December 2007

Fitting the December 2007 API and mashup data to the model using nonlinear regression yields N_{50} = 8.24, α = 0.570 and β = 1.14. Figure 4 plots the actual data along with fitted data from the model. Although the model fits the data very well, it would be misleading to conclude that the mashup distribution has a long tail in the same sense as books and movies do. On the contrary, the parameters indicate that API use by mashups has a shorter tail than book sales on all three dimensions, as well as a shorter tail than movie viewership on two dimensions, N_{50} and β.

This result is consistent with the frequency data reported earlier. Recall from Section 4.1 that the head of the mashup distribution is top-heavy, with the top 20% of the APIs capturing 95% of the mashups, a far greater share than the 80% suggested by the Pareto rule. Moreover, the tail flattens out quickly at the bottom end, with 51% of the APIs excluded because they were not used by any mashups at all.

5 Social Network Statistics

The impression conveyed by the analysis so far is that the most popular APIs are vastly more important than the rest in terms of their contribution to the mashup ecosystem. While it is true that a small set of APIs account for the majority of mashups (with the top 10 responsible for well over 50% of the observed mashup links), a closer look at the network structure reveals a subtler but perhaps equally important role for the less popular APIs in the ecosystem.

In this section, we focus on the API affiliation network. Investigating the structure of API affiliations enables us to explore patterns of innovation by mashup creators. Since each link in the affiliation network represents a different pair of APIs used by one or more mashups, this analysis sheds light on the ways in which mashup creators combine APIs to create applications with novel functionality.

We computed a set of network metrics commonly employed in the social network literature: mean degree, normalized degree, network density, characteristic path length and clustering coefficient. These metrics were calculated for each of the 9 quarterly network snapshots, and are summarized in Table 1.

Table 1. API affiliation network metrics by quarter

Date	*Mean Degree*	*Normalized Degree*	*Network Density*	*Characteristic Path Length*	*Clustering Coefficient*
Dec-05	1.152	0.0127	0.0162	2.355	0.320
Mar-06	2.971	0.0175	0.0272	2.284	0.500
Jun-06	4.901	0.0231	0.0415	2.228	0.399
Sep-06	5.279	0.0189	0.0344	2.206	0.395
Dec-06	6.171	0.0176	0.0329	2.243	0.418
Mar-07	9.170	0.0230	0.0395	2.223	0.458
Jun-07	9.155	0.0200	0.0350	2.237	0.448
Sep-07	8.929	0.0172	0.0344	2.240	0.428
Dec-07	8.488	0.0144	0.0281	2.282	0.414

5.1 API Affiliation Metrics

The degree of a network node is the number of connections the node has with others. In the API affiliation network, the degree indicates the number of other APIs that are used in common with a given API by one or more mashups. Examining the *mean degree* of the network over time, we see a steady increase from 1.152 mashups per API in December 2005 to a maximum of 9.170 in March 2007, followed by a plateau and a slight downward trend for the remainder of the year.

The *normalized degree* is the mean degree divided by the maximum possible degree (the total number of nodes minus one), providing a measure of network connectivity that controls for the growth of the network. Over the period of the study, the normalized degree varied much less than the mean degree, ranging from about 0.013 to 0.023.

The *network density* is the ratio of the number of actual links in the network to the total number of possible links that would exist if all nodes were directly connected to each other. Hence, it is another measure of network connectivity, i.e., the extent to which APIs are connected to each other by mashups. The network density fluctuated substantially over the study period, from about 0.016 to 0.045, but without a clear trend.

The *characteristic path length* (CPL) is the expected distance along the shortest path between any two nodes in the network. After a slight drop between December 2006 and March 2006, it remained fairly stable (between about 2.2 and 2.3).

The *clustering coefficient* (CC) measures the extent to which network nodes tend to form groups with many internal connections but few connections leading out of the group. Like the CPL, it remains nearly constant after March 2006 (between about 0.40 and 0.45).

5.2 Small-World Analysis

The characteristic path length and clustering coefficient are the key statistics used to identify networks with the *small-world property* coined by Watts and Strogatz [12]. The concept of a small-world network was inspired by Stanley Milgram's famous experiment in which randomly selected individuals in Nebraska and Kansas were able to forward letters to a target in Boston through an average of only six intermediaries.

In the formalization of the concept proposed by Watts and Strogatz, a network has the small-world property if its characteristic path length is similar to that of a random network with the same density despite having a much larger clustering coefficient. In a random network, the expected CPL is approximately $\ln n / \ln k$ and the expected CC is approximately k, where n is the number of nodes and k is the mean degree of the network. For the API affiliation network in December 2007, $CPL_{rand} = 2.98$ and $CC_{rand} = 0.0144$, while the actual CPL was 2.28 (even lower than CPL_{rand}) and the actual CC was 0.414 (almost 30 times higher than CC_{rand}). The API affiliation network thus easily qualifies as a small-world network. Further investigation shows this to be true throughout the two-year sample period.

What does it mean for the API affiliation network to have the small-world property? Loosely speaking, nodes in a small-world network are more closely connected than one would expect based on their density and clustering. In the context of the mashup ecosystem, this suggests that APIs with very different functionality (e.g., mapping, audio search, and news feeds) are more likely to be connected through mashups (e.g., an application that shows famous places from popular songs, and one that finds news stories on popular artists) than one might otherwise expect. Although we caution against reading too much into this finding, it is an encouraging sign that the mashup ecosystem has generated a substantial level of novelty and surprise.

5.3 Importance of Peripheral APIs

To better understand why the API affiliation network has the small-world property, we repeated the analysis on the affiliation network formed by a subset of the most popular APIs. To construct the subset, we selected APIs that were used by at least 5 mashups in December 2007. These 152 APIs comprised 57% of the APIs with at least one mashup, or about half of the APIs that played an active role in the mashup ecosystem.

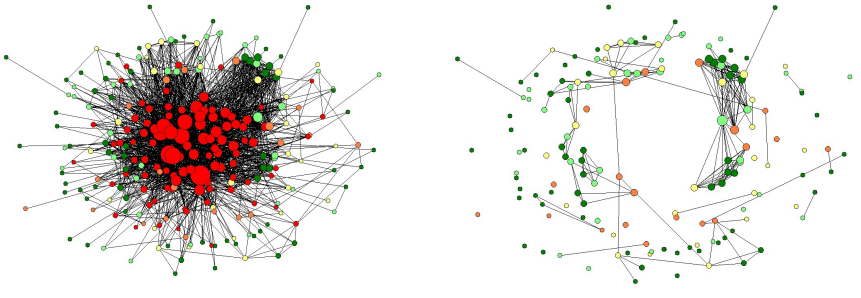
Omitting the less popular APIs from the network reveals their important role in giving rise to the small-world property of the mashup ecosystem as a whole. As Table 2 indicates, the affiliations among most popular APIs barely qualify as small-world. Their clustering coefficient (0.446) is similar to the full set, but only 3.02 times that of a comparable random network, compared to 28.78 for the full set. Despite this comparatively low level of clustering (which intuitively ought to reduce path lengths), the CPL of the subset affiliation network is longer than CPL_{rand}, indicating a relative absence of the distinctive short paths associated with the small-world property.

Table 2. Small-world metrics for a subset of the most popular APIs (December 2007)

APIs	n	k	CC	CC_{rand}	$\frac{CC}{CC_{rand}}$	CPL	CPL_{rand}	$\frac{CPL}{CPL_{rand}}$
Full set	590	8.49	0.414	0.0144	28.78	2.28	2.98	0.765
Popular	152	22.42	0.446	0.1475	3.02	1.99	1.62	1.233

The role of the less popular APIs in forming these short paths can be seen in Figures 5a and 5b. These visualizations were generated using NetDraw's spring embedding algorithm, which locates pairs of nodes with the shortest path lengths closest to each other. The full network is shown in Figure 5a; the network with the most popular APIs omitted is shown in Figure 5b. The nodes are colored according to the number of mashups for each API: red nodes represent the most popular 152 APIs with 5 or more mashups, while orange, yellow, light green and dark green represent APIs with 4, 3, 2 and 1 mashups respectively. APIs with no mashups were omitted. The size of each node corresponds to the degree of each API, with larger nodes indicating APIs with higher degrees.

In Figure 5a, the most popular APIs form the core of the network, and are highly interconnected through the mass of black lines. The less popular APIs in orange, yellow, light and dark green, form rings around the core, indicating that they are further away in terms of path length. These peripheral APIs mostly have connections to the APIs in the core, forming clusters around them. Figure 5b shows clearly that there are few direct connections between less popular APIs.



(a) (b)

Fig. 5. (a) Full set of API affiliations. (b) Affiliations of peripheral APIs.

This additional analysis highlights the structural differences between the more and less popular APIs in the network. This is consistent with the qualitative findings of Section 3, in particular that APIs can be roughly segmented into two types: core platform APIs, which tend to form hubs since they can be used with many other APIs, and peripheral APIs, which tend to be more specialized in function (e.g., supplying a particular type of raw data) and less attractive for reuse. The structure of the affiliation network suggests that innovation (i.e., the appearance of mashups that create new affiliations) is concentrated in two areas: within the core set of platform APIs, and between the platform APIs and the periphery.

6 Conclusion

The growth of the mashup ecosystem has attracted media attention and fueled high expectations. In 2006, commentators called it "incredibly ripe for innovation" [18] and saw "numerous forces combining to make the mashup ecosystem 'explode'" [19]. However, the ProgrammableWeb.com data revealed a more modest growth pattern. (In fact, the growth rate declined toward the end of our sample period and into 2008.) Moreover, as the highly concentrated distribution of API popularity makes clear, simply releasing an API is no guarantee that mashup creators will build on it. These factors should encourage ecosystem participants to think carefully about the roles they want to play, and how to succeed in these roles.

This paper was motivated by the desire to help inform such thinking with empirical data from the mashup ecosystem. Our analysis was intended to be exploratory rather than conclusive. It shows that mashup developers do not simply mix and match APIs arbitrarily to create new mashups. Instead, they tend to build on a small subset of platform-like APIs (which become very popular and densely interconnected) while drawing less frequently from a much wider range of APIs that perform more specialized functions. By far the most popular of the platform APIs is Google Maps, which was used by 48% of the mashups in our sample.

Despite the apparent dominance of the most popular APIs, we find a significant role for the ecosystem's less popular APIs as well. Many of these peripheral APIs are involved in mashups that bring together novel combinations of functionality, thus creating new links in the API affiliation network. We view the structural richness of this network as a sign of innovative activity in the mashup ecosystem.

There are many limitations in our analysis. In particular, although ProgrammableWeb.com was the most comprehensive repository of API and mashup data available to us, it relies extensively on data reported by API providers and mashup creators. Both groups have incentives to participate (there is no cost to list an API or mashup, and both stand to benefit from the visibility provided by the site), but there are undoubtedly APIs and mashups in existence that are not registered.

There is also much more analysis that could be done. This paper took a first cut at describing the characteristics of the mashup ecosystem through graphical visualization and quantitative investigation of its network structure. Future research can build on these findings to develop a more rigorous theoretical framework for explaining the patterns we observed and predicting how the ecosystem will evolve. We hope this and subsequent work will help Web 2.0 participants make sound strategic choices about designing and releasing APIs, and enable mashup creators to innovate more effectively by recombining these APIs in compelling new ways.

Acknowledgments. Many thanks to John Musser of ProgrammableWeb.com for providing API access to the site, and to Darshan Santani for his generous help with data extraction and cleanup.

References

1. O'Reilly, T.: Web 2.0: Compact Definition? (2005), http://radar.oreilly.com/archives/2005/10/web-20-compact-definition.html
2. Wikipedia: Mashup, http://en.wikipedia.org/wiki/Mashup_web_application_hybrid
3. Cho, A.: An Introduction to Mashups for Health Librarians. JCHLA 28, 19–22 (2007)
4. Kulathuramaiyer, N.: Mashups: Emerging Application Development Paradigm for a Digital Journal. JUCS 13, 531–542 (2007)
5. O'Brien, D.S., Fitzgerald, B.F.: Mashups, Remixes and Copyright Law. INTLB 9(2), 17–19 (2006)
6. Goodman, E., Moed, A.: Community in Mashups: The Case of Personal Geodata (2006), http://mashworks.net/images/5/59/Goodman_Moed_2006.pdf
7. Jackson, C., Wang, H.J.: Subspace: Secure Cross-domain Communication for Web Mashups. In: 16th International World Wide Web Conference, pp. 611–620 (2007)
8. Liu, X., Hui, Y., Sun, W., Liang, H.: Towards Service Composition Based on Mashup. In: 2007 IEEE Congress on Services, pp. 332–339 (2007)
9. Hinchcliffe, D.: Is IBM Making Enterprise Mashups Respectable? (2006), http://blogs.zdnet.com/Hinchcliffe/?p=49
10. Barabási, A.-L., Albert, R.: Emergence of Scaling in Random Networks. Science 286, 509–512 (1999)
11. Anderson, C.: The Long Tail (2004), http://www.wired.com/wired/archive/12.10/tail.html
12. Watts, D.J., Strogatz, S.H.: Collective Dynamics of "Small-World" Networks. Nature 393, 409–410 (1998)
13. Wasserman, S., Faust, K.: Social Network Analysis: Methods and Applications. Cambridge Univ. Press, Cambridge (1994)
14. Borgatti, S.P.: NetDraw: Graph Visualization Software. Analytic Technologies (2002)
15. Adamic, L.A.: Zipf, Power-Laws, and Pareto: A Ranking Tutorial, http://www.hpl.hp.com/research/idl/papers/ranking/ranking.html
16. Brynjolfsson, E., Hu, Y.J., Smith, M.D.: From Niches to Riches: Anatomy of the Long Tail. Sloan Mgmt. Rev. 47(4), 67–71 (2006)
17. Kilkki, K.: A Practical Model for Analyzing Long Tails. First Monday 12(5) (2007)
18. Berlind, D.: Mashup Ecosystem Poised to Explode (2006), http://blogs.zdnet.com/BTL/?p=2484
19. Hinchcliffe, D.: The Web 2.0 Mashup Ecosystem Ramps Up (2006), http://web2.socialcomputingmagazine.com/the_web_20_mashup_ecosystem_ramps_up.htm

The Changing Role of IT Departments in Enterprise Mashup Environments

Volker Hoyer[1,2] and Katarina Stanoevska-Slabeva[1]

[1] University of St. Gallen, Institute for Media and Communications Management, 9000 St. Gallen, Switzerland
[2] SAP Research, 9000 St. Gallen, Switzerland
{volker.hoyer,katarina.stanoevska}@unisg.ch

Abstract. A new paradigm, known as Enterprise Mashups, implicates a shift concerning the service development and consumption process: end users combine and reuse existing Web-based resources within minutes to new applications in order to solve an individual and ad-hoc business problem. In such democratized operational environments, the role of IT departments is changing. They are no longer solely responsible for developing or installing business applications. Instead, end users in the business units compose their own operational environment in a collaborative manner. This paper analyses and discusses challenges and the changing role of IT departments toward service intermediaries by leveraging the St. Gallen Media Reference Model (MRM).

Keywords: Enterprise Mashups, St. Gallen Media Reference Model.

1 Introduction and Motivation

Tradition problems between IT departments and business units, such as a low service transparency, low reaction time, lack of customer orientation, and poor quality of IT support, are no longer accepted, a fact which is demonstrated by the tendency to build up independent IT resources with business units [1]. In addition, the growing relevance of information centric and situation applications to address the individual and heterogeneous needs of end users [2], leads to a new generation of Web-based applications, known as Enterprise Mashups. By empowering users in the business units with no programming skills to create collaboratively the own operational environment, IT departments are under pressure to justify their existence on the one side and to increase the efficiency and effectiveness of the IT service infrastructure on the other side [1],[2].

However, an analysis of the implications of usage Enterprise Mashups environments is missing. The goal of this short position paper is to identify and analyse the challenges in context of Enterprise Mashup environments - in particular regarding the role of IT departments. The remainder of this paper is structured as follows: Chapter two clarifies the terminology used in context of the Enterprise Mashup paradigm and contrasts the development model against traditional Service-Oriented Architectures. Based on the St. Gallen Media Reference Model

G. Feuerlicht and W. Lamersdorf (Eds.): ICSOC 2008, LNCS 5472, pp. 148–154, 2009.

(MRM), chapter three outlines the shifting role of the IT department regarding from the MRM layers community, interaction, service, and infrastructure. Finally, chapter four closes the paper with a summary and an outlook to further research.

2 Related Work

2.1 Enterprise Mashups – Definition and Characteristics

An Enterprise Mashups is a Web-based resource that combines existing resources, be it content, data or application functionality, from more than one resource in enterprise environments by empowering the actual end users to create and adapt individual information centric and situational applications [3]. Thereby, Enterprise Mashups focus on the UI integration [4] by combining the philosophy of Service-Oriented Architecture (SOA) and approaches of End User Development (EUD) [3].

Table 1. Service-Oriented Architecture versus Enterprise Mashups

Criteria	**Service-Oriented Architecture**	**Enterprise Mashups**
Time-to-value	Many weeks, months, or even years	Minutes, hours or days
Developer Profile	IT department	Business units (limited programming skills); small teams or individuals
Integration Layer	Application Integration	Focus on UI Integration
Development Phases	Well defined, following agreed-to schedule (although with frequent schedule overruns)	No defined phases or schedules; focus on a good-enough solution to address an immediate need
Functional Requirements	Defined by limited number of users, IT needs to freeze requirements to move to development, requirement creep often caused by changing business needs	As requirements change, Enterprise Mashups usually changes to accommodate business changes; Enterprise Mashups encourages unintended uses
Nonfunctional Requirements	Resources allocated to address concerns for performance, availability, and security; robust solutions	Little or no focus on scalability, maintainability, availability, etc.
Testing	By IT with some user involvement	By users through actual uses

With the assistance of a layer concept, the relevant components and terms can be structured in an Enterprise Mashup Stack [3] consisting of the elements resources, widgets and Mashups. **Resources** represent actual contents, data or application functionality. They are encapsulated via well-defined public interfaces (Application Programming Interfaces; i.e. WSDL, RSS, Atom, CSV, XML, etc.) allowing the loosely coupling of existing Web-based resources - a major quality of SOA. The layer above contains **widgets** which are responsible

for providing graphical and simple user interaction mechanism abstracting from the underlying technical resources. Users can combine and configure such visual widgets according to their individual needs, which results in a **Mashup**.

Key driver of the Enterprise Mashup paradigm is the lightweight composition style by reusing existing building blocks in new ways - getting value out of prior investments. The mass collaboration is an additional driver. The willingness of users to offer feedback to the Mashup creator who may be unaware of problems or alternative uses, directly contributes to the adoption of the Mashup and can foster its ongoing improvement. Rating, recommending, tagging, or sharing features for the different Enterprise Mashups layers, support the collaborative reuse of existing knowledge to solve daily business problems [3].

To understand the changing development environment, table 1 summarizes the findings of a desk research [5],[6],[3],[2] and experiences taken from first implementations of domain specific Mashups with the SAP Research Rooftop Mashup prototype. The comparison of the traditional development approach and the Enterprise Mashups paradigm indicates the changing environment.

2.2 Enterprise Mashup Platforms

Driven by the consumer-oriented industry, various Mashup tools and platforms were developed in the last two years. According to the classification of [7] and by applying the Enterprise Mashup Stack [3], we can distinguish between Mashup platforms and widget editors on the one dimension and between the enterprise and consumer target group on the other dimension. Because this paper focuses explicitly on enterprise requirements, we narrow the following discussion on enterprise-oriented platforms:

- **Widget Platforms.** Widget platforms and editors allow to compose heterogeneous Web-based resources ("'piping"') and to put a visual face on the technical resources. Well known widget tools are Yahoo Pipes, Microsoft Popfly, SAP Research Rooftop, or IBM Damia. The composition results of these tools (sometimes they are also called as data Mashups) can be consumed by desktop environments (Vista Gadgets, Yahoo Widgets), mobile devices (Apple iPhone, Nokia Symbian), or Mashup platforms.
- **Mashup Platforms.** In contrast to widget platforms, Mashup platforms address the actual end user with no programming skills. By adding new widgets from a catalogue and by connecting their input and output parameters ("'wiring"'), end users are empowered to customize their individual operational environment. Examples for Enterprise Mashup platforms are IBM Mashup Center/ Infosphere Mashup Hub (based on the research projects IBM QEDWiki and IBM Mashup Hub), JackBe Presto Edge, Serena Mashup Suite, or the Open Source project EzWeb.

Both types of tools have in common the lightweight composition style and the integrated community features to share, rate, or recommend a mashable component (resource or widget) similar to electronic markets.

3 St. Gallen Media Reference Model for Enterprise Mashup Environments

In order to structure the analysis, we revert to the St. Gallen Media Reference Model (MRM)[8]. Due to the similarities to electronic markets as identified by [9] and also indicated in the section before, we leverage the St. Gallen Media Reference Model which has its roots in electronic markets. It provides a framework for specifying IT infrastructures and has already been applied in different contexts successfully (i.e., modeling electronic markets[8] or m-commerce applications[10]). Under the term medium, we understand platforms based on information and communication technologies, i.e., communication spaces of "social interaction which allow the participant to meet and which embed them in a common physical, logical, and socio-organizational structure" [11].

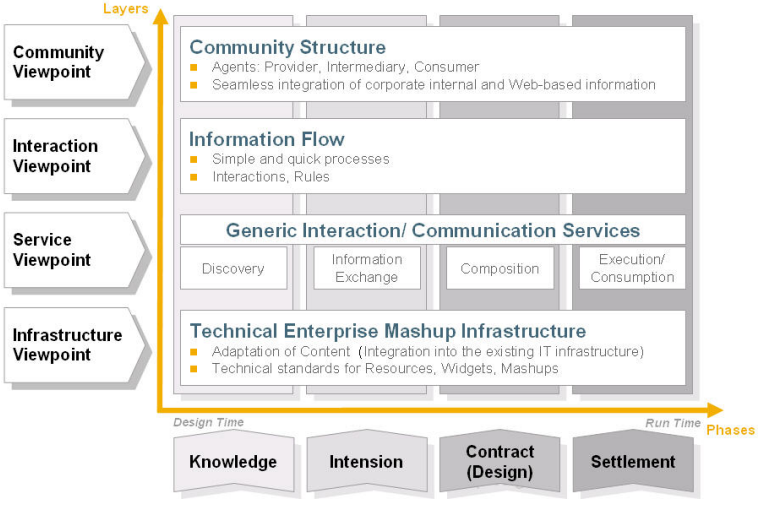


Fig. 1. St. Gallen Media Reference Model for Enterprise Mashup Environments

The media reference model provides guidelines for how to build a medium based on information and communication technology by guiding the process of requirement evaluation and by identifying the required services. It provides four layers to structure the different successive interaction goals of the participating agents. The *community view* describes the participating agents and the organizational structure. The *interaction (process) view* refers to the procedural description of the interaction events. It models the community view requirements by means of the *service view* which provides the necessary services for carrying out the described process steps in the interaction view. Finally, the *infrastructure view* contains communication protocols and standards which comprise the groundwork for the implementation of services.

In addition, the MRM identifies four phases. First, the *knowledge phase* is which information about offered services and knowledge and the media platform

itself is acquired. Second, the *intention phase* in which agents signal their intentions in terms of offers and demand. Third, the *contract (design) phase* where agents compose their individual workspace and finally the *settlement phase*, in which agents execute the designed applications, using the platform's settlement services offered for this purpose. Figure 1 depicts a first version of a reference model for Enterprise Mashup environments. Following each component of the model is described briefly focusing on the changing role of IT departments.

3.1 Community View

A critical success factor for Enterprise Mashups is a broad potential user group, familiar with the technology and willing to use it in their daily operational environment. In general, Mashups are developed in very small user groups or by individuals. The possible interactions and tasks of the agents can be described by the following model: A *provider* develops and publishes a Mashup element via a *intermediary*, where a *consumer* can find it and subsequently may bind to the provider. In contrast to SOA, users from the business units don't just interact as consumers. They are able to create their own Mashup element and provide it to the community. Besides the traditional provider role, the IT department takes over the intermediary role. It monitors continuously the parameters (such as availability and response latency) and provides performance metrics and other evaluation results (rating, tagging, recommending, etc.) which may be used by consumers to select a Mashup element [3]. In addition, the richness of Enterprise Mashups applications are based on combining seamlessly corporate internal with Web-based information sources. So Web providers have to be integrated into the Enterprise Mashup community to publish their value added Mashup elements.

3.2 Interaction View

Figure 2 depicts a simplified process describing the interaction between the main roles covering the four MRM phases. According to the findings of section two, the

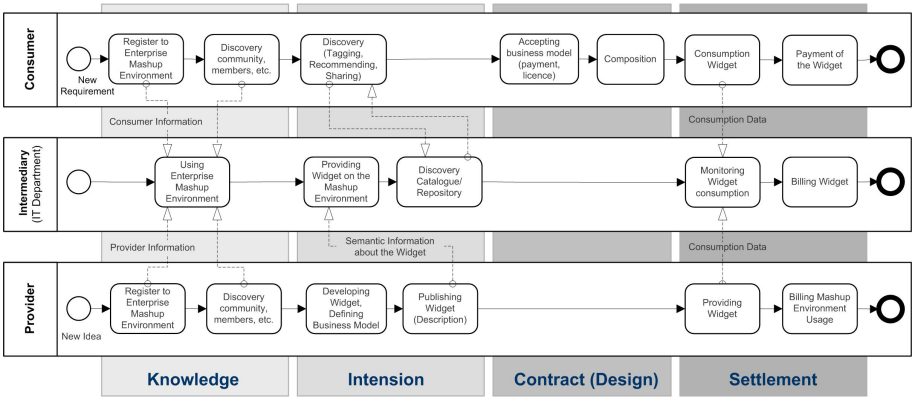


Fig. 2. Simplified interaction process between Enterprise Mashup agents

process itself has to be simple and quick as possible - in particular for the users from the business units. They focus on solving daily business problems in the sales or accounting department and not on creating or adapting their operational environment. The IT department is responsible to hide the complexity and to support the actual end-users towards a service intermediary.

3.3 Service View

The growing number of available mashupable elements requires adequate *discovery* concepts for retrieval purposes. According to the user context (profile, preferences, social network it belongs to) relevant services are presented to the users who are able to select the right Mashup element. *Sharing of information*, experiences and knowledge with the community is a key driver for Enterprise Mashups. Besides the default semantic annotations (functional and non-functional qualities) defined by the provider, consumers are able to tag, recommend, or rate the elements. By creating a folksonomy, essential a bottom-up, organic taxonomy, consumers are empowered to organize the available elements. The *composition* takes place both on the resource layer (piping) and on the widget (wiring) layer. In reference to the UNIX shell pipeline concept, the *piping* composition integrates heterogeneous resources. Aggregation, transformation, filter, or sort functions adapt and mix the underlying resources. The visual composition of input and output ports on the widget layer is called *wiring*. In contrast to piping that requires skills in programming and data standards, wiring can be done by users without special IT skills. Good enough solutions within minutes lead to a converging design and run time (*execution*). From consumer perspective, no deployment exists. They design their operational environment and execute it immediately. For IT departments, the execution means providing support for administrating, monitoring, and accounting the consumed Mashup elements.

3.4 Infrastructure View

In contrast to existing applications (i.e., MS Excel or MS Access) created and managed by business units to address ad-hoc requirements, the infrastructure of Enterprise Mashups environments are managed by the corporate IT department. Business units are empowered to integrate easily their local resources or backend systems into the environment. However, wide accepted standards (widget, Mashups), protocols for the visual composition (piping or wiring), or accounting methods are still missing in existing Enterprise Mashup environments [7].

4 Conclusion

The aim of the paper is the analysis of the changing role of IT departments towards service intermediaries in Enterprise Mashup environments. In order to achieve this, the main terms related to Enterprise Mashups were defined. By applying the St. Gallen Media Reference Model, we structure the analysis to identify the challenges implicated by the democratized environments.

However, the model serves only as a starting point and framework for further research focusing on the different views. In frame of the EU funded research project Fast and Advanced Storyboard Tool (FAST) [12], we are currently developing an infrastructure and the relevant services for the creation of widgets. By means of various real-world industry scenarios, we will analyse and observe in detail the relationship between IT departments and business units. In addition, we will use the designed reference model of this paper to analyse the economic benefits of the Enterprise Mashups paradigm.

Acknowledgments. This paper has been created closely to research activities during the EU-funded project FAST (INFSO-ICT-216048) [12].

References

1. Zarnekow, R., Brenner, W., Pilgram, U.: Integrated Information Management. Applying Successful Industrial Concepts in IT. Springer, Berlin (2006)
2. Cherbakov, L., Bravery, A., Goodman, B., Pandya, A., Bagget, J.: Changing the corporate IT development model: Tapping the power of grassrots computing. IBM System Journals 46(4) (2007)
3. Hoyer, V., Stanoevska-Slabeva, K., Janner, T., Schroth, C.: Enterprise Mashups: Design Principles towards the Long Tail of User Needs. In: IEEE International Conference on Services Computing (SCC), vol. 2, pp. 601–602 (2008)
4. Daniel, F., Matera, M., Yu, J., Benatallah, B., Saint-Paul, R., Casati, F.: Understanding UI Integration. A Survey of Problems, Technologies, and Opportunities. IEEE Internet Computing 11(3), 59–66 (2007)
5. Cherbakov, L., Bravery, A., Pandya, A.: SOA meets Situational Applications: Changing Computing in the Enterprise (2007), http://www.ibm.com/developerworks/
6. Janner, T., Canas, V., Hierro, J., Licano, D., Reyers, M., Schroth, C., Soriano, J., Hoyer, V.: Enterprise Mashups: Putting a face on next generation global SOA. In: Tutorial at the 8th Int. Conf. on Web Information Systems Engineering (2007)
7. Hoyer, V., Fischer, M.: Market Overview of Enterprise Mashup Tools. In: Proceedings of the 6th Int. Conf. on Service Oriented Computing (ICSOC) (2008)
8. Schmid, B., Lindemann, M.: Elements of a Reference Model for Electronic Markets. In: Proceedings of the 31st Hawaii Int. Conf. on System Sciences (HICSS) (1998)
9. Legner, C.: Is there a Market for Web Services? - An Analysis of Web Services Directories. In: Proceedings of the 1st International Highlight Workshop on Web APIs and Services Mashups (2007)
10. Stanoevska-Slabeva, K.: Towards a Reference Model for M-Commerce. In: Proceedings of European Conference on Information Systems (ECIS 2003) (2003)
11. Schmid, B.: The Concept of Media. In: Proceedings of the Fourth Research Symposium on Electronic Markets, pp. 77–90 (1997)
12. FAST: EU Project, INFSO-ICT-216048 (2008), http://fast.morfeo-project.eu/

The Mashup Atelier

Cesare Pautasso and Monica Frisoni

Faculty of Informatics
University of Lugano (USI)
via Buffi 13, 6900 Lugano, Switzerland
cesare.pautasso@unisi.ch, monica.frisoni@lu.unisi.ch

Abstract. Can mashups be used to make high school students interested in studying computer science? To answer this question, we have designed the mashup atelier. The goal of this experimental lecture is to make students realize that the Web is not only a medium for passively consuming information but it can be actively reprogrammed as they see fit. The atelier introduces the topic of Web 2.0 Mashups to students without any formal pre-existing computer science education. After giving the atelier several times, we report on the results of a student evaluation survey showing that, if supported with right kind of mashup tools, creative students can become very productive developing interesting mashups in a short timeframe. The feedback we gathered from the students can also be used to improve existing mashup languages and tools, with the ultimate goal of understanding what makes them intuitive and fun to use.

1 Introduction

Mashup [1] development targets the so-called *long tail* of application development, enabling end-users to compose Web services and Web data sources by themselves to fulfill their own specific requirements [2]. Ideal mashup development tools should require very little upfront training and exhibit a gently-sloped learning curve [3,4]. Using such tools, end-users having very little programming experience can quickly become productive and build their own mashups [5].

To check whether mashup development is indeed a suitable target for end-user software engineering, we designed an experimental lecture called "The Mashup Atelier". In this paper we present the feedback we gathered by observing end-users as they build mashups and asking them about their experience attending the lecture. We contribute this valuable feedback to help improving existing mashup tools and languages that also target end-users without programming skills.

The mashup atelier was originally designed in the context of an ongoing national initiative to make high school students aware of the computer science discipline and to awake their interest in pursuing a computer science degree. The goal of the atelier was to use mashups to convey to the students the idea that the Web is not only an interactive medium for consuming information, but it is also a new kind of medium where they can become active participants [6]. By

G. Feuerlicht and W. Lamersdorf (Eds.): ICSOC 2008, LNCS 5472, pp. 155–165, 2009.

showing the students how to program the Web by building mashups, we hoped to unleash their creativity and inspire them to learn more about informatics and computer science.

In this paper we show that – given the right kind of tools – indeed it is possible for end-users to develop simple but interesting mashup applications without previous programming experience. We also show that mashup development can be a useful approach for attracting the interest of young students to the computer science discipline.

The rest of this paper is organized as follows. In Section 2 we define the structure of the mashup atelier and outline the questionnaire that was handed out to the students. We list some of the mashups that were developed during the atelier in Section 3. Section 4 contains the results of the survey and the feedback we gathered by observing the students. We discuss related work in Section 5 and draw some conclusions in Section 6.

2 Methodology

The mashup atelier is structured in two parts and lasts approx. 3 hours in total. During the first part, students are given some theoretical background on the inner workings of the Web and are presented with many examples of Web 2.0 services. This part concludes by introducing the notion of mashup. To do so, several example mashups are shown to students with the promise that they would be soon ready to develop similar ones.

The main part of the atelier (2 hours) is practical. Students attending it have access to a PC pre-configured with the mashup development environment. The practical mashup development work starts with a 20 minute step-by-step tutorial. Afterwards the students may work independently, first to implement a few exercises (or challenges), then to create their own mashups.

During the atelier we interacted with the students, guiding them, answering their questions and observing their progress. Thus, the students were intentionally not left alone to try to develop mashups on their own. On the contrary, students were allowed to collaborate and to share their work with one another. Our goal was not to test each student's abilities, but to create a positive working atmosphere. Towards the end of the atelier we gave the students an evaluation questionnaire (Table 1) to fill out and were able to collect answers from all participants.

The students worked with the Microsoft PopFly mashup development environment [7]. This tool was chosen due to several reasons. First of all, the tool does not require to be installed locally (apart from the requirement of having the Microsoft Silverlight runtime [8]). This way, students can start developing mashups during the atelier and then – if they used their own MSN accounts to log in – they are able to continue working on them from home. Additionally, this free tool is rather mature and stable. It provides a large library of reusable mashup building blocks, which can be explained by referring the students to the concept of function ($y = f(x)$), known to students from their high school

Table 1. Summary of the Feedback Questionnaire

1. Are you a member of a social networking site? (Yes/No) If Yes, which ones?
2. Do you know how to program? (Yes/No) If Yes, with which languages?
3. Did you already know what is a 'Mashup' before attending the atelier? (Yes/No)
4. Did you know how to use Microsoft Popfly before attending the atelier? (Yes/No)
5. What was your impression of the mashup development tool? Why?
6. Was the mashup tool intuitive? (Yes/No) Why?
7. What did you like most about the mashup tool?
8. What did you dislike most about the mashup tool?
9. Will you keep using the mashup tool in the future? (Yes/No/Maybe) Why?
10. Overall, are you satisfied about the mashup atelier? (Yes/No) Why?

math. It features a very quick design-run-test cycle and also enables users to share mashups by publishing them on their homepages. We also thought that the rich visual environment (with 3D animations) provided by Popfly would make it appealing to young high school students.

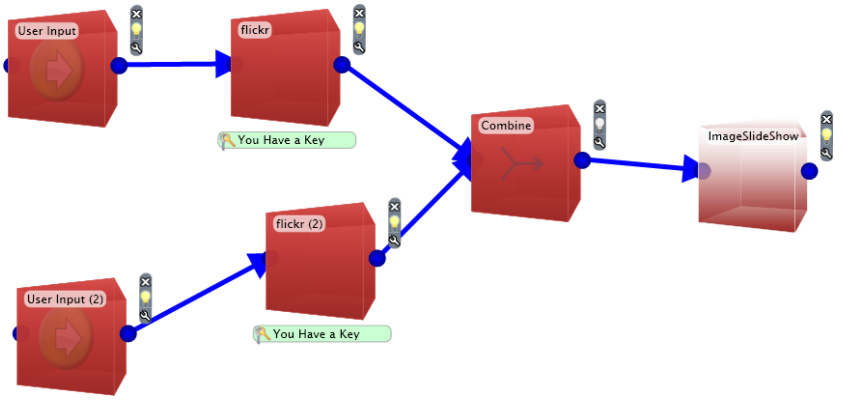
3 Mashup Examples

In this section we briefly describe some example mashups developed by the students attending the atelier. This helps to give an idea of the difficulties involved and of the complexity of the resulting mashups that were developed in the limited time available (1 hour and 30 minutes).

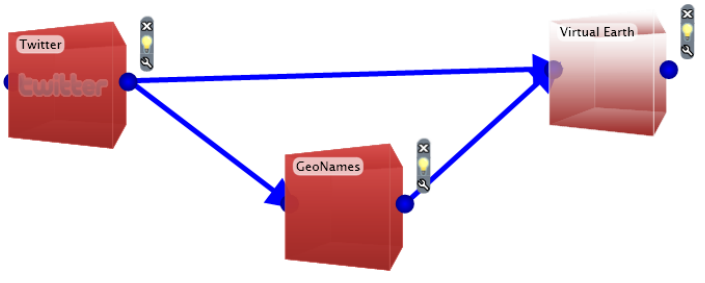
All students were initially guided through the development of a simple example mashup that would display a set of images retrieved from the flickr service. This simple example can be implemented in Popfly using two blocks: the first to search flickr for pictures and the second to display the results. Such warmup exercise helped students to understand how to configure the input parameters of a block and how to draw a connection between two blocks in order to specify data flow. During this first exercise, students were also given a walkthrough the mashup development environment so that they learned how to design, run, test, and modify their mashups.

Starting from this basic example, students were assigned the following exercises to grow the complexity of their mashup and discover more useful blocks and features of the PopFly environment:

- Display the pictures of a predefined search term on a map – this would require the mashup to fetch from flickr the geo-tagging data associated with the pictures and pass this information to the 'VirtualEarth' display block.
- Let the user enter the search terms during the mashup execution – this requires to add an additional user input block and to correctly link it with the flickr search block. Some students testing this extended version of the mashup realized that an incorrect linkage would always result in the same pictures being displayed, as the data they entered would not flow between the boxes at run-time. This shows that support for debugging is an important feature of end-user software engineering tools.



(a) Slide show with multiple sets of pictures



(b) Twittervision

Fig. 1. Mashup Examples

– Display pictures extracted from two different sites (i.e., Yahoo! Images and Flickr) – this exercise would require students to learn how to merge multiple data streams into a single one using the combine operator provided by PopFly. Students with good observation skills would then inquire on why pictures of the second set are not appended to the first and the two sets are simply interleaved by the combine block.

After working through these exercises of increasing complexity, the students had some time left for browsing the block library and creating their own mashups. Several students extended the previously described mashup collecting images from multiple sources into a mashup to retrieve and mix multiple sets of images (e.g., cats and dogs) from the same source (Figure 1a). Inspired by the twittervision.com mashup, some students were able to access a twitter feed, geo-code the location of its entries and display them on a map. To do so, they had to discover the 'Twitter' block from the standard PopFly library and learn how to correctly use it in conjunction with the 'GeoNames' block also found in the library (Figure 1b). One student was able to use the 'RSS' block to aggregate multiple technology-related newsfeeds and display them into a widget that could be embedded into his homepage. Another student successfully attempted to publish a mashup with the daily horoscope onto her facebook profile.

4 Survey Results

4.1 Background

All 43 students (29 male and 14 female) attending the four editions of the Mashup atelier held in September 2008 at the University of Lugano, in the Italian-speaking part of Switzerland, participated in the survey. Their age distribution (between 16 and 21 year-old) is shown in Figure 2. Since their native language is Italian, both the feedback questionnaire and the answers we collected were in Italian and had to be translated to be included in this paper.

All students declared their familiarity with the Web and knowledge about some common Web 2.0 services. Half of the students also used one or more social networking sites (e.g., 10 Netlog, 9 MySpace, 6 Facebook, 5 WebTI, 4 MSN). We were surprised to find out that at least five students are also running their own blog.

Figure 3 shows which programming languages (including HTML) the students have been in contact with. 26 declared not to have programming experience. If we exclude HTML, then only 9 students had previously learned at least one programming language (the most popular one being Visual Basic). We can thus identify two sub-groups of students, the ones without programming experience and the ones with some programming experience.

None of the students we interviewed knew the term 'Mashup' before the atelier. Also, no student had previously been in contact with the Microsoft PopFly mashup development environment.

4.2 Overall Impression

Figure 4 shows the number of answers to the question concerning the overall impression of the students after working with the mashup tool for two hours. Multiple answers were possible. The majority of the students found PopFly useful and interesting. Only 4 students complained about some limitations and 1 student found it useless. Only one student admitted he did not know how to evaluate the tool given he had never seen one like it.

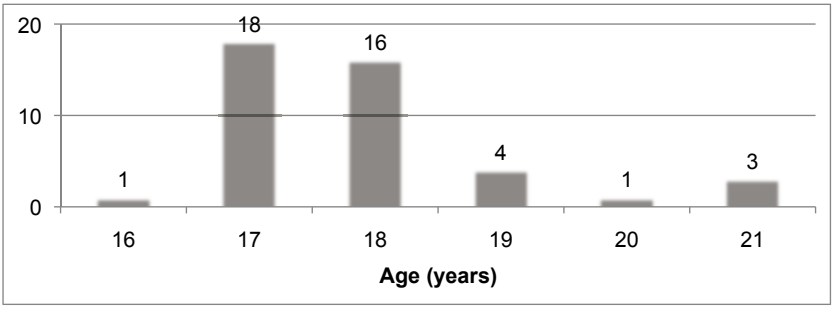


Fig. 2. Age distribution of the students attending the Mashup Atelier

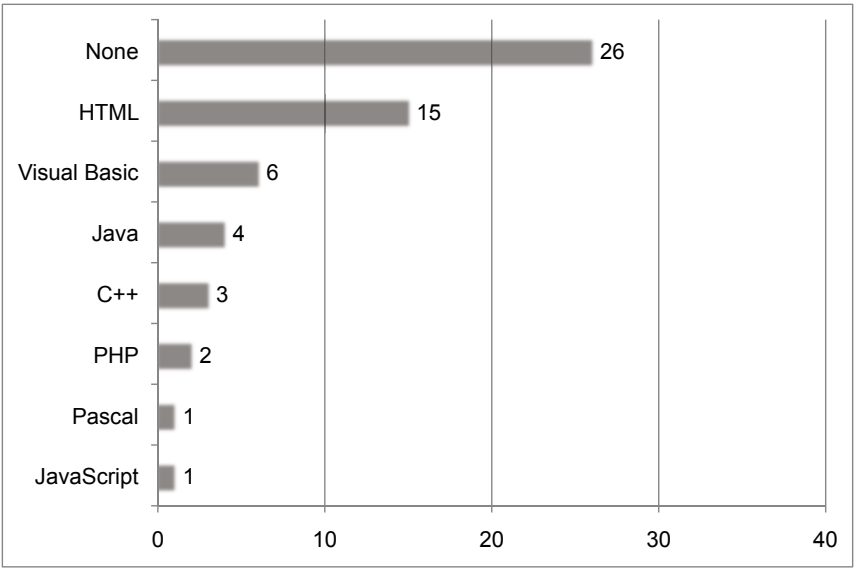


Fig. 3. Existing Experience with Programming Languages

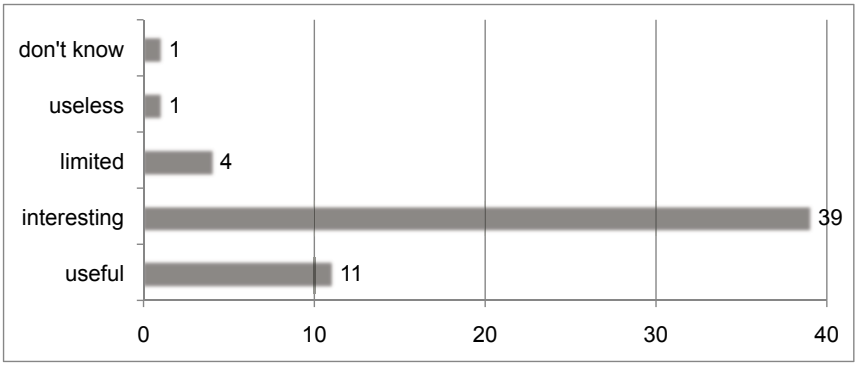


Fig. 4. What was your impression of the mashup tool?

From the positive side, the students that found the tool useful grasped its potential to "save valuable time, since all data can be put in one page". Students also found it an interesting idea to make multiple Web pages "overlap" when various data sources are combined. Many students enjoyed using their mashup to browse the flickr picture database and were also attracted by the various display blocks at their disposal. One was positively impressed because he "did not think it would have been possible to create a mashup so easily".

Some of the negative opinions were caused by glitches in the tool. One student had problems saving his homepage. One could publish the mashup but

complained that the PopFly logo is embedded into the published mashup. Another student with some previous knowledge about programming complained that the tool lacks support for interactive debugging. One student pointed out that he found the tool both interesting because of its visual block-based programming paradigm, but also limited because he would not have enough knowledge to be able to add his own personalized block to the library.

4.3 Was the Mashup Tool Intuitive?

As shown in Figure 5, the large majority (including 24 students without programming skills) of the students agreed in finding PopFly an intuitive mashup development environment.

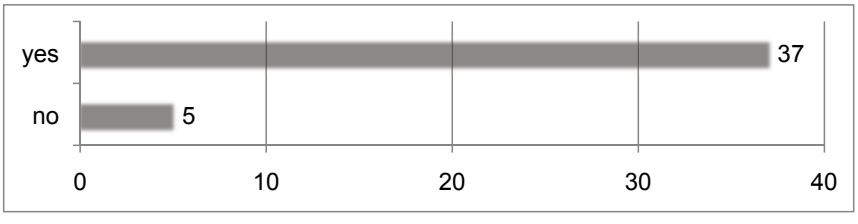


Fig. 5. Was the mashup tool intuitive?

The students giving a positive answer motivated their choice for many reasons. Some were related to the properties of the development environment. "The environment supports fast trial and error". "It is fun to use". "To work with it, is enough to try out some blocks from the library and figure out how to connect them". "Blocks have tooltips with descriptions of their functionality". "Once you understand how to connect the boxes, it is easy". "Even if I am not a computer expert, I could more or less finish all of the exercises". Other students also commented on the language itself. "It does not use a technical language". "The language uses symbols rather than words". "The language basic connection mechanism is not too complex and anyone can use it thanks to its clear graphical notation". "It works even if you have never seen a programming language". "The graphics look visually simple and the available commands are not too complicated." "Even if it is in English, it is rather understandable and simple".

The 5 negative answers were explained as follows. One student admitted that he "needed to ask the teacher for help", despite his knowledge of Pascal and Visual Basic. Two students with knowledge of HTML recognized that the tool "requires good computer science skills" and "it is not very interactive". The two students without programming knowledge stated that "It was the first time I used it", and "It looks rather complicated".

4.4 Positive Feedback

In general, the students liked learning about the notion of mashup, and the idea of creatively mixing together information coming from various Web pages.

The students were enthusiastic to have a sort of control to some known and powerful Web applications, such as "Google Maps" or "Virtual Earth", and to easily have access to pictures from "Flickr" and "Yahoo! Images". The idea that they could embed them into their personal web page, and show to their friends what they could create, made them appreciate the mashup philosophy and let them discover an alternative way to use the Web.

The students liked the following specific features of the mashup development tool. On the one hand, they appreciated how quickly they could play with the blocks of the library, experiment with their mashups and build visually appealing slide shows. On the other hand, a few students also liked the rich graphical notation and the search feature of the library browser tool. The ability of easily gaining control over a certain Web data source or display widget was found very positive by many students. They enjoyed searching for pictures and displaying them using different kinds of visualizations. They were fascinated by the fact that data and images coming from different well-known Web services could be merged together into a new mashup application designed by them.

Some students were also very excited about the possibility of sharing their mashup and publishing them on their homepage. After learning about this feature, one student quickly invited all of her friends present in the classroom to join her new PopFly social network and forced them to "become a fan" of her mashup.

4.5 Negative Feedback

23 students also gave some constructive criticism from their experience with the mashup development tool. Only 2 students wished the tool was available in the Italian language.

The block library attracted many suggestions for improvement, showing that it is one of the most important features of a mashup development tool. Some students felt overwhelmed and confused by the size of the library with hundreds of reusable building blocks. Others were satisfied of having so much choice at their disposal. Thus, whereas it is hard to argue in favour of a smaller or a larger block library, its accessibility becomes very important. To help them getting started, we gave them a list of 10 useful blocks and suggested they would ignore the user-contributed blocks (since they may not always work). The automatic suggestion feature was also used by the students to filter the block library and make it display only blocks that could somehow be connected to the current one. The results of this feature were not always clearly understood, as they did not distinguish the blocks that could be used to provide input from the ones that could be used to consume the output. Also, some pointed out that the descriptions associated with the blocks and their configuration parameters were not always easy to understand.

After becoming familiar with the block library, many students realized that the set of mashups they could build is limited by the available blocks they could find and understand how to use. Thus, they would not be able to build a mashup unless they could find a suitable set of blocks to implement the required

functionality. Thus, PopFly supports very well a bottom-up composition scenario where users are guided building mashups starting from existing blocks. However, it only offers partial support for top-down design and decomposition. This became evident to some students, since they realized that they would not be able to implement their own blocks to supply the missing functionality (For example, to scrape information out of a Web page). This is due both to their lack of JavaScript programming skills, but also to the lack of documentation of the PopFly block development API. The students that dared to switch to the advanced configuration view of a block were simply lost facing the low-level code implementing the block.

Some students were also annoyed by the requirement of obtaining and entering a registration key, as this would break their creative exploration flow. Thus, it would be convenient to provide pre-registered blocks for demo purposes that can be later on configured with the proper credentials. Four students also complained about the limitation concerning the maximum number of display blocks that could be added to their mashup. Others were not successful in using the display widgets to show their own pictures or could not find a block to play music.

Regarding the usability of the graphical notation, several students – at first – were confused by the positioning of the input/output connectors of a block. Without an explicit explanation they did not independently grasp the convention of using the connector on the right-hand side to represent the output and the one on the left to represent the input of the block. Whereas this is a common notational convention of many visual languages, for complete beginners it remains an arbitrary convention that needs to be explained. Once this was clarified, drawing arrows between boxes simply required them to learn the correct mouse click/drag sequence.

Another limitation pointed out by the more technically savvy students is related to the requirement for having the Silverlight extensions installed both for developing mashups but also for displaying them once they are shared and published on their own hompage. Also, they were disappointed that once their mashup was embedded into their personal Web page, they would not be able to hide the Microsoft PopFly logo.

4.6 Will You Keep Using PopFly in the Future?

Also in this case, as summarized in Figure 6, the majority of students answered that they would be interested in further usage of PopFly at home.

The positive replies were motivated because the students were curious to find out more about the tool's potential applications. They found it interesting, useful, and even "cute and fun". Two students emphasized the possibility of embedding mashups into their own web pages and they planned to use the tool to build interactive photo albums, only if they could find out how to build mashups feeding them their own pictures and personal data.

The uncertain replies were due to some difficulties that were encountered during the atelier, lack of time to explore the tool in depth, and a vague: "you never know if it will become useful someday".

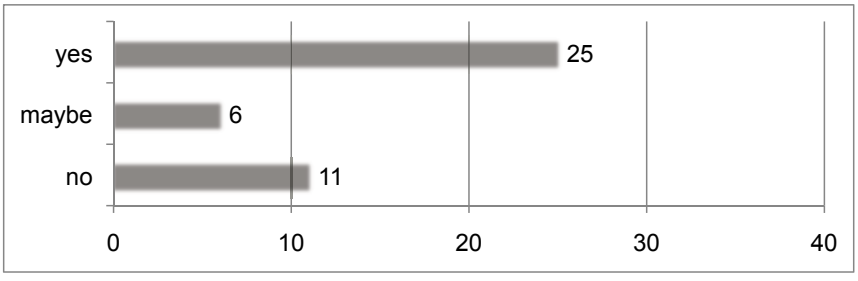


Fig. 6. Will you keep using the mashup tool in the future?

The negative replies did not give much insight on how the tool could be improved, as this group of 11 students seemed to be satisfied with browsing existing Web sites and participating in existing social networking tools. They were not interested in further exploration and did not find an immediate need for the tool. One stated that: "he does not normally use the computer for this kind of things".

5 Related Work

The potential for using mashups for educational purposes was first discussed in [5]. In the same context, [9] is a more recent paper documenting the usage of mashup development tools with teenagers. A comprehensive survey on end-user programming languages and environments can be found in [4].

Whereas most existing research on mashup development environments shares the goal of making it fast and easy to serendipitously reuse [10] and compose existing Web services and Web data sources into a mashup, only a few contributions explicitly target inexperienced end-users. For example, the Mashmaker tool [11], is presented as a user-friendly mashup environment based on a functional programming language. Also, Marmite [12] addresses the needs of end-users with its highly interactive development environment. We did not considering using these tools in the mashup atelier due to their limited availability.

6 Conclusion

This paper reports on our initial experiences with the Mashup Atelier, an interactive, project-based lecture for introducing young high school students with the topic of Web 2.0 Mashups. This experimental atelier was designed with three main objectives. The first was about testing whether young students with no programming experience are capable of building mashups within a short timeframe. The second was to gather feedback from the student impressions to benefit future generations of "intuitive" mashup development environments and languages. The third was to see if mashup development could be effectively used to get young students interested in learning more about computer science. Overall,

the mashup atelier received very positive feedback and a high degree of student satisfaction (41 out of 43 students were satisfied with it, while the remaining 2 did not answer the question). It is still early to measure how many of the high school students who attended will choose to enroll in a computer science degree program. Nevertheless, from the results of our survey we observe that exposing students to mashup development using a visual and interactive tool such as Microsoft PopFly was very useful to get their attention.

Acknowledgements

This work is partially supported by the Informatica08 initiative. The authors would also like to thank Mauro Prevostini for his excellent logistical support with the atelier organization and Monica Landoni for her help in improving an early draft of the student feedback questionnaire.

References

1. Wikipedia: Mashup (web application hybrid), http://en.wikipedia.org/wiki/Mashup_web_application_hybrid
2. Hoyer, V., Stanoesvka-Slabeva, K., Janner, T., Schroth, C.: Enterprise mashups: Design principles towards the long tail of user needs. In: Proc. of IEEE International Conference on Services Computing (SCC 2008), pp. 601–602 (July 2008)
3. Rode, J., Bhardwaj, Y., Pérez-Quiñones, M.A., Rosson, M.B., Howarth, J.: As easy as "Click": End-user web engineering. In: Lowe, D.G., Gaedke, M. (eds.) ICWE 2005. LNCS, vol. 3579, pp. 478–488. Springer, Heidelberg (2005)
4. Kelleher, C., Pausch, R.: Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. ACM Comput. Surv. 37(2), 83–137 (2005)
5. Lamb, B.: Dr. Mashup or, Why Educators Should Learn to Stop Worrying and Love the Remix. EDUCAUSE Review, 7 (2004)
6. Jhingran, A.: Enterprise information mashups: integrating information, simply. In: VLDB 2006: Proceedings of the 32nd international conference on Very large data bases, Seoul, Korea, pp. 3–4 (2006)
7. Microsoft: Popfly, http://www.popfly.ms/
8. Microsoft: Silverlight, http://www.microsoft.com/silverlight/
9. Yardi, S.: From functional to fun: End user development for teenagers. In: Proc. of the IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC 2007), pp. 272–274 (September 2007)
10. Vinoski, S.: Serendipitous reuse. IEEE Internet Computing 12(1), 84–87 (2008)
11. Ennals, R., Gay, D.: User-friendly functional programming for web mashups. In: ICFP 2007: ACM SIGPLAN International Conference on Functional Programming - SIGPLAN Not., vol. 42(9), pp. 223–234 (October 2007)
12. Wong, J., Hong, J.: Marmite: end-user programming for the web. In: CHI 2006 extended abstracts on Human factors in computing systems, Montreal, Quebec, Canada, pp. 1541–1546 (2006)

The Reverse C10K Problem for Server-Side Mashups

Dong Liu and Ralph Deters

Department of Computer Science
University of Saskatchewan
Saskatchewan, Canada
dong.liu@usask.ca, ralph@cs.usask.ca

Abstract. The original C10K problem [1] studies how to provide reasonable service to 10,000 simultaneous clients or HTTP requests using a normal web server. We call the following problem the reverse C10K problem, or RC10K — how to support 10,000 simultaneous outbound HTTP requests running on a web server. The RC10K problem can be found in scenarios like service orchestrations and server-side mashups. A server-side mashup needs to send several simultaneous HTTP requests to partner services for each inbound request. Many approaches to improving the performance and scalability of HTTP servers can be applied to tackle the original C10K problem. However, whether these approaches can tackle the reverse C10K problem needs to be verified. In this paper, we discuss the RC10K problem for server-side mashups, and propose a design that takes advantage of advanced I/O, multithreading, and event-driven programming. The results of analysis and experiments show that our design can reduce the resource requirements by almost one order of magnitude with the same performance provided, and it is promising to tackle the RC10K problem.

Keywords: HTTP, Mashup, Scalability, Performance, Client, C10K, RC10K.

1 Introduction

AJAX (Asynchronous JavaScript and XML) and mashup applications are efficient interfaces for consuming published services on the web. An AJAX or mashup page gets data from one or more services hosted on different servers [2]. If a mashup is generated by on-demand code on client agent (client-side mashup), requests are typically sent out by an API like XMLHttpRequest (XHR) of JavaScript. XHR acts as an HTTP client in such scenarios. The server hosting those AJAX and mashup pages is not responsible for requesting data from partner services, and all those computations are carried out on the client by code-on-demand. The situation is different when the HTTP request tasks of a mashup is executed on the server (server-side mashup, SSM for short), and the SSM server is responsible for fetching data from partner services by sending outbound HTTP requests through broker clients. The conceptual structure of an SSM server is shown in Fig. 1.

G. Feuerlicht and W. Lamersdorf (Eds.): ICSOC 2008, LNCS 5472, pp. 166–177, 2009.

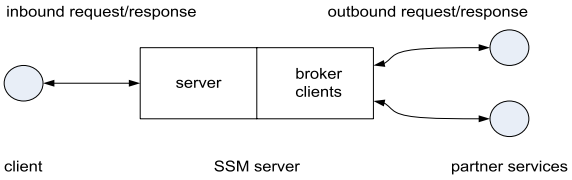


Fig. 1. The conceptual structure of an SSM server

HTTP brings two constraints [3] to the clients running on an SSM server:

(1) The basic message exchange pattern is request-response.
(2) An established connection is required for message transportation.

The constraints means a mashup server needs to maintain at least one connection for each outbound service consumption, and perform at least one request-response transaction on that connection. In order to reduce the service time of an inbound request, an SSM will launch parallel outbound requests. This yields simultaneous outbound connections and message exchanges. If there are N simultaneous inbound requests for an SSM server, and each request results in C parallel outbound requests, there will be CN simultaneous outbound connections and active outbound HTTP requests in the worst case. If an SSM server needs to handle several thousand simultaneous requests, then the server can have about $10,000$ or more simultaneous outbound HTTP requests to deal with. How an SSM server can support that many simultaneous outbound HTTP requests is what we called the reverse C10K problem, or RC10K for short.

The original C10K problem [1] studies how to provide reasonable service to $10,000$ concurrent clients using a normal server. Many approaches to improving the performance and scalability of HTTP servers can be applied to tackle the original C10K problem [1,4,5]. However, whether and how these approaches can tackle the RC10K problem of an SSM server are still open questions. This paper proposes a client design that adapts the approaches for the original C10K problem to this RC10K problem. The evaluation shows that our approach can effectively improve the scalability of an SSM server, and tackle the RC10K problem. The rest of this paper is structured as following. Section 2 discusses the approaches addressing the original C10K problem. An architectural design of SSM server's broker client is presented in Section 3. Section 4 evaluates the design by experiments. Related work is discussed in Section 5. Section 6 is the conclusions and future work.

2 The C10K Problem

The essential of the C10K problem is how to support a large number of inbound TCP connections and how to serve the concurrent inbound requests, which leads to two key design decisions of HTTP servers: I/O and concurrency strategy.

2.1 I/O

The input/output models can be divided into two groups: basic I/O and advanced I/O. Basic I/O is synchronous and blocking. An I/O operation is synchronous if the thread initializing the I/O operation cannot switch to other operation until the I/O operation is finished. A function or method is blocking if it does not return until its execution either successfully finishes or encounters an error. There are three ways to make advanced I/O [6]. The first approach is to construct a loop to keep trying an I/O option while catching I/O errors until it succeeds. This approach is called polling and it wastes CPU time. The second approach is I/O multiplexing uses `select()`-like system functions. Most operating systems support `select()`, and it is also supported by Java VM 1.4 and later versions. The descriptors of connections can be registered on a `selector`, which calls `select()` to check if there is any I/O event for each of the descriptors. So a thread initializing an I/O operation can delegate the job to a `selector` and switch to other job. Note that `select()` is blocking until one of the registered descriptors has an event or timeout happens. The third approach is asynchronous I/O (AIO), which is both asynchronous and non-blocking. Both I/O multiplexing and AIO enable a server to use very few threads to handle many concurrent connections. In practice, I/O multiplexing and AIO are applied for developing scalable servers [1].

2.2 Concurrency

Servers can roughly be classified into two categories, single-threaded and multithreaded, according to their concurrency model. Multithreading is favoured in many servers as a means to dealing with simultaneous requests, increasing the degree of concurrent processing, and making use of multiprocessors e.g. multicore processors [7]. This improves the performance of the platform compared to single-threaded implementations [8].

HTTP 1.1 [3] introduced persistent connections, which enables a client to send multiple requests through the same connection. Persistent connections improved the keep-alive connections in HTTP 1.0 [9]. By using persistent connections, the time to open or close connections can be saved and the number of parallel connections needed by a fixed number of requests can be reduced [10]. A connection can have two different behaviours — idle for a long period or busy with back to back requests. Polling of Web 2.0 applications is a typical cause of the second situation. It is tricky to optimize the usage of threads when allocating threads to connections in a multithreading server. Resources will be wasted if a thread is allocated for each connection and the connections are often idle. The strategy of allocating a thread for each request will also waste resource if the request has to be hold waiting for other messages or events and the thread cannot be switched to other job.

For a multithreading server programmed in Java or .Net, the number of threads is a measure of allocated resources, since it is directly related to CPU and memory consumption. It has been observed on .NET applications [11] and Java EE applications [12,13] that the response time of an application will increase

with respect to the number of active threads in an application server until the server is overloaded. Therefore, the concurrency model needs to:

(1) limit the maximum numbers of threads that can be spawned in the server, and
(2) keep the number of active threads as few as possible.

The first requirement results in a bounded thread pool. For the second requirement, the threads need to be programmed in an event-driven fashion, which yields a hybrid server architecture [14,15]. There are two different strategies for the second requirement — one is to make the waiting thread idle until the event for its job comes, and the other is to switch the thread to another active job and resume the current waiting job later on the corresponding events.

3 Design for the RC10K Problem

It is straightforward to apply the I/O and concurrency strategies of the server to the broker client. However, the client has two new problems that the server does not have. One problem is how to reuse the available connections in order to save connection open and close time and keep the concurrent opened connection number below the maximum number allowed on partner server. The other problem is how to embedded the client code into a mashup application easily.

A solution of the first problem is to introduce a component for connection management to the client. We propose a modular design shown in Fig. 2. Our design is inspired by the design of Jetty server [16], whose details can be found in Section 5. The client contains a connector that is responsible for connecting to a server and transporting messages to and from it. A thread pool provides worker threads to perform the client jobs. A connection manager creates, registers, and reuses connections. The connector and thread pool of the server can be reused by the client. The SSM server and the broker client can share the same thread pool instance for easy coordination of resource allocation.

The second problem can be addressed by introducing a message exchange object or structure, which has conceptually four components: destination, socket connection, request message, and response message. Its major behaviours are the state transitions during message exchanging and the actions driven by the transition events.

The state transitions of an HTTP client are shown in Fig. 3. A socket connection is initially closed when created by a client, and the connection will be established when a `SYN+ACK` from the server side is received. The client may retry connecting for several times before giving up if the connection attempts are either rejected or timeout. HTTP requests then can be sent through the established connection. The client will wait for the response when the request is completely sent. When the response comes, the client will first parse out the header then the message body. When the whole response is completed, the connection will become idle and ready for another request to be sent. Timeout may happen when the client is waiting for or getting a response. The client may try to resend the request several times before giving up.

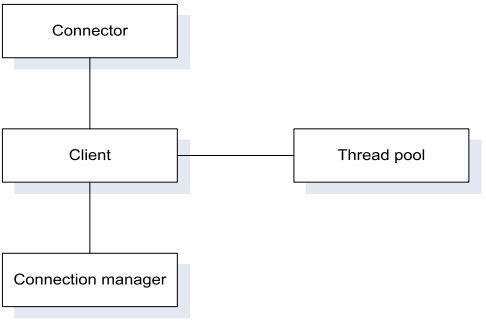
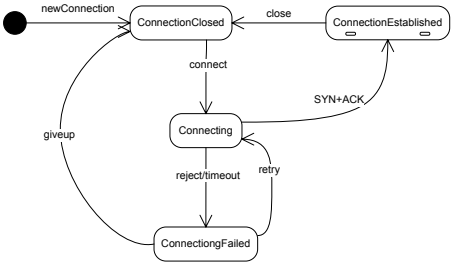
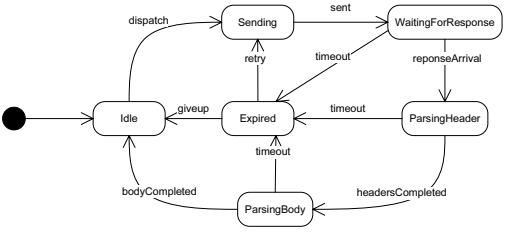


Fig. 2. The architecture of the proposed HTTP client



(a) State transitions of transportation.



(b) Detailed state transitions of the composite state `ConnectionEstabished`.

Fig. 3. A state transition diagram of an HTTP client

The important events and corresponding potential actions described in Fig. 3 are listed in Table 1. Some events happen on socket I/O level, and therefore the responsibility of capturing them can be allocated to the connector. These events can be programmed as the methods of exchange object. So a mashup application just needs to create an exchange object that implements the required methods. The major purpose of combining event-driven programming with multithreading is to enable asynchronous processing. Asynchronous processing means that, the processing of a request can be suspended when waiting for responses of outbound requests, and it can be resumed when those responses come or timeout happens. Asynchronous processing can save computation resource effectively.

Table 1. The important events and associated potential actions for an HTTP exchange

Event	Potential actions
On connection established	update the connections for the current IP socket address, prepare request message to be sent
On connection failed	handle the failure
On connection idle	update the connections for the current IP socket address
On waiting for response	switch the thread to other job
On headers completed	process the headers or wait for the body
On body completed	locate a thread for current request and process the message
On expired	handle the expiration

4 Evaluation

We want to verify if the design described in Section 3 can lower resource demand for the same workload compared with other design options. The number of active threads is used as the indicator of resource consumption. First we derive an analytic result and then use experiments to verify the analysis.

4.1 Analysis

How many threads are required in an SSM server for N concurrent requests? The number will be a linear function of N, the number of concurrent inbound requests, as the following equation.

$$n = N \times (1 + c) \,, \tag{1}$$

where 1 represents the demand by the thread allocated for the inbound request, and c is the number of threads required by outbound HTTP requests. Furthermore, c can be calculated by

$$c = \frac{\sum_{i=1}^{C} D_i}{S} \,,$$

where C is the number of outbound service consumptions for each inbound request, D_i is the time demand for ith outbound service consumption, and S is the average service time of an inbound service request. From Equation 1, there are two ways to decrease n — reducing either D_i or 1. D_i will become smaller by switching the threads to other jobs while it is waiting for outbound responses. Similarly, if a thread initially allocated to an inbound request can be switched to another job by asynchronous processing, the 1 can become smaller, and the average thread number will be

$$n = N \times (\frac{D}{S} + c) \,, \tag{2}$$

where D is the time demand for inbound service request. In order to have an inbound or outbound service request suspended and resumed later, the platform needs to support continuation-like mechanism [17,18].

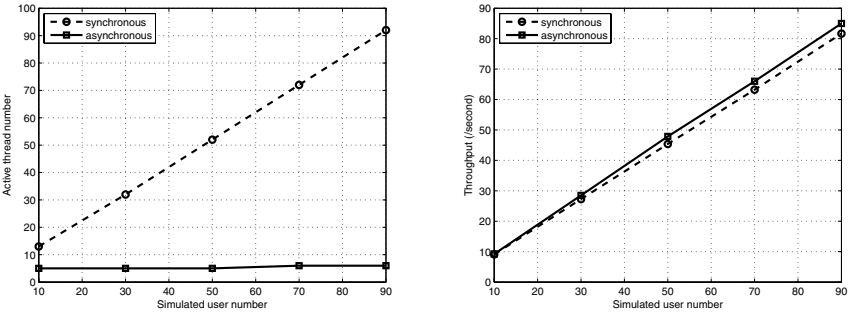
4.2 Experiments and Results

In order to evaluate the design presented in Section 3, we carried out a series of experiments. The mashup server in the experiments are programmed in Java based on Jetty 7.0[1]. Note that implementation details are not the focus of this paper and implementations vary for various programming paradigms and languages.

The first aspect to be evaluated is how event-driven programming and asynchronous processing can optimize the usage of computation resources. We use two machines in this experiment, machine A is running Window XP SP3 and Java SE runtime environment 1.6 on 3.2GHz P4 CPU with Hyper-threading[2] and 2GB RAM, and machine B is running Window XP SP3 and Java SE runtime environment 1.6 on dual 3.2GHz Xeon CPU's with Hyper-threading and 2GB RAM. Both of them have the TcpIP parameter `TcpTimedWaitDelay`[3] set as 60 seconds. The two machines are connected with a router, and the connection speed is 100Mbps. JMeter[4] is running on machine A to simulate end users. The think time of simulated users is set to zero in order to make the number of concurrent requests in the intermediary service as close to the number of simulated users as possible. We chose JMeter for load generation because JMeter can control the exact number of concurrent running clients. A server-side mashup is running on machine B, and it consumes two other partner services to generate responses. The broker client uses `select()` type I/O. The two partner services are hosted by Jetty running on machine B as well. One partner service spends about 0.5 seconds for each request, and the other about one second. Both of them reply with tiny payloads. The resource requirements for running the two partner services are low enough for not interfering the mashup's performance. Fig. 4 shows the active thread number and throughput as functions of the number of simulated users in synchronous and asynchronous processing modes.

As shown in Fig. 4(b), the throughputs for synchronous and asynchronous processing are almost the same because most of the response time is spent on getting response from the partner services. There is a distinction of active thread number between two processing models shown in Fig. 4(a) The thread number for synchronous processing is about linear with respect to the number of simulated users. Note that the number of simulated users is very close to the number of simultaneous active requests due to zero think time and low network latency. This fits Equation (1) very well. On the contrary, the thread number for asynchronous processing is likely constant, which does not seem to accord to Equation (2) at first glance. In fact, the thread demand is still increasing with the user number in this case, but it is very slow because $\frac{D}{S}+c$ in Equation (2) is very small. Table 2 lists the values of D, S, and $\frac{D}{S}$ in the experiments. Asynchronous processing

[1] See http://www.mortbay.org/jetty/
[2] See http://www.intel.com/technology/platform-technology/hyper-threading/index.htm
[3] See http://support.microsoft.com/kb/314053
[4] See http://jakarta.apache.org/jmeter/index.html



(a) Active thread number (b) Throughput

Fig. 4. Active thread number and throughput as functions of the number of simulated users in synchronous and asynchronous processing modes

Table 2. The average time demand and service time for intermediary service request

Users number	10	30	50	70	90
D **(millisecond)**	0	1	1	1	2
S **(millisecond)**	1014	1012	1012	1015	1014
D/S	0	0.0010	0.0010	0.0010	0.0020

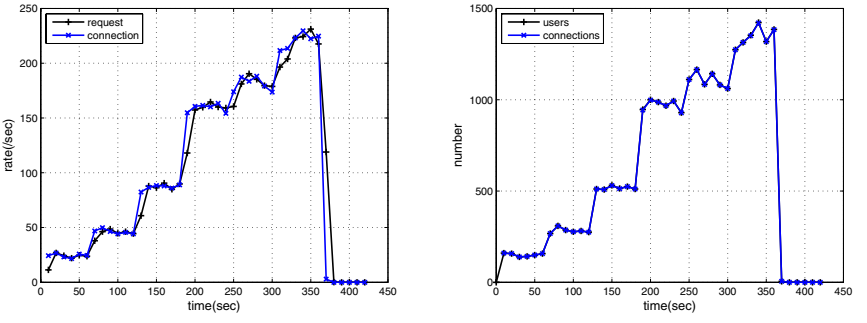
can drop the resource requirements of intermediary services dramatically when $\frac{D}{S}$ is far less than 1.

The second aspect to be evaluated is whether our design is sufficient for the RC10K problem, or how close it can be pushed to the target. In this experiment, the mashup server is running on machine B. The tested mashup consumes 3 partner services 2 times for each inbound request. The partner services are hosted by YAWS[5] on three other machines running Linux 2.6.24-19-server, and their hardware is exactly the same as machine B. Tsung[6] running on machine A is used to generate the load. All the machines are connected through 100 Mbps LAN. Tsung is choosed in this scenario because it can generate workload of high arrival rate, and therefore is able to simulate an open network environment. Machine B is specially tuned for the large number of inbound and outbound TCP connections. The TcpIP parameter `MaxUserPort` is set as 65534, `MaxFreeTcbs` as 10000, and `MaxHashTableSize` as 8192. The JVM is tuned for heap size and thread stack size, specially `-Xss64k -Xms1024M -Xmx1024M -XX:PermSize=256M -XX:MaxPermSize=256M` is used in this experiments.

Each consumption of a partner service takes averagely 5 seconds. This time is deliberately set to be large compared to normal services in order to obtain a large number of concurrent outbound connections for a certain arrival rate. Each test

[5] See `http://yaws.hyber.org/`
[6] See `http://tsung.erlang-projects.org/`

is composed of several phases back to back, and each phase takes one minute. Fig. 5 shows the throughput and concurrent users of the mashup server in one of the tests. The inter-arrival time (seconds) changes along different phases from 0.04 to 0.02, 0.01, 0.005, 0.004, and 0.003. The inter-arrival time is decreased gradually in order to warm up the SSM server and reach its maximum capacity gracefully.



(a) Throughput. (b) Concurrent users.

Fig. 5. The throughput and concurrent users along different phases of a test

The largest number of concurrent users that has been reached is about 1400 when the server still operates normally. When the inter-arrival time decreases to 0.002 seconds, the server becomes unstable. The largest number of concurrent outbound connections is about 8400 (1400×6). Although it is less than $10,000$, it is very close to the target. The same tests were performed on WSO2 mashup server 1.5.1 on which a mashup with the same logic was deployed. The WSO2 mashup server is overloaded when the inter-arrival time reached 0.1 seconds, that is, the server cannot even support 50 concurrent inbound requests and 300 concurrent outbound requests. The details about WSO2 mashup server can be found in Section 5.

5 Related Work

As discussed in Section 2, there have been many research and development activities related to the original C10K problem or the scalability issues of HTTP servers. There are few research literature addressing the scalability issues of HTTP client that can be seen in the RC10K problem. We expect there will be more and more research work related to this topic with the development of mashup and web service applications.

XMLHttpRequest (XHR) is probably the most popular HTTP client interface currently used for Web-based service consumption. XHR's interface was specialized by W3C [19] and its implementations vary a lot in different Web browsers.

Each XHR object requires an event listener that is normally a callback function describing the actions to be triggered on certain events. The function is called every time the `readyState` changes. `readyState` can be in one of the five states, namely request unsent, `open()` success, response header received, loading response body, and response done. The XHR also provides accesses of request header, request body, response status, response header and response body. XHR is a perfect example for high-level message exchange interface design. However, XHR specification does not address the aspects of connection and thread management.

HTTPCLIENT[7] is a Java-based open-source project at Apache. HTTPCLIENT depends on HttpCore NIO extensions[8] to support non-blocking I/O (NIO) and event-driven programming. HTTPCLIENT is used in some SSM servers like WSO2 Mashup Server[9]. HTTPCLIENT has a component for connection management, but does not provide thread pool. Without a thread pool, it will be difficult to reuse the available idle threads.

WSO2 Mashup Server is a platform that uses JavaScript as the language of representing and programming mashups. In other words, it exposes JavaScript functions as services. WSO2 Mashup server provides several hosted objects that ease common mashup operations like fetching feeds, scrapping web pages, and sending either HTTP or SOAP requests to other services. The WSRequest object[10] mimics the XHR interface, and is able to perform both synchronous and asynchronous requests. Due to the limitation of JavaScript and the underlying Mozilla Rhino JavaScript engine, the response of an asynchronous request can only be caught by using a `wait()` to hold the requesting thread, which make the asynchronous request consume more resources and run slow.

The tested SSM server of proposed architecture is developed on the basis of Jetty 7.0 and especially its Servlet 3.0 features[11]. Jetty is an open-source web server implemented in Java. Jetty provides dynamic content support through Servlet and JSP technologies. We leverage the client code base of Jetty with shared thread pool and asynchronous processing for the evaluation. Note that the official Jetty project focuses on HTTP server, and the client is just an 'extra' part. More efforts are needed to improve the client code base, and make an SSM server out of it.

6 Conclusions

The scalability and performance of the broker client in an SSM server directly affect server scalability and performance. To date, the scalability issues of HTTP clients have been overlooked in research. Although many approaches have been studied for improving HTTP servers' scalability like the C10K problem, whether

[7] See http://hc.apache.org/httpclient-3.x/
[8] See http://hc.apache.org/httpcomponents-core/index.html
[9] See http://wso2.org/projects/mashup
[10] See http://wso2.org/project/mashup/1.5.1/docs/wsrequesthostobject.html
[11] See http://jcp.org/en/jsr/detail?id=315

those approaches are effective for HTTP clients is still an open question. We formulate the reverse C10K problem in this paper, and propose an architectural design of the client that uses advanced I/O, multithreading, and asynchronous processing in order to tackle the RC10K problem.

The evaluation shows that our design can reduce resource requirements by almost one order of magnitude in order to achieve the same performance compared with other designs using synchronous processing. The evaluation also shows that the 10,000 simultaneous outbound connections is feasible in a normal web server setup and virtual machine environment like JVM. Currently, we are investigating how much better other languages like Erlang[12] can perform in the RC10K problem due to its features like light-weight process, no memory-sharing, and built-in message-passing.

References

1. Kegel, D.: The c10k problem. Web (September 2006), http://www.kegel.com/c10k.html
2. Ort, E., Brydon, S., Basler, M.: Mashup styles, part 1: Server-side mashups. Web (May 2007), http://java.sun.com/developer/technicalArticles/J2EE/mashup_1/
3. The Internet Society: Hypertext transfer protocol – http/1.1. Web (June 1999), http://tools.ietf.org/html/rfc2616
4. Darcy, J.: High-performance server architecture. Web (August 2002), http://pl.atyp.us/content/tech/servers.html
5. Barish, G.: Building scalable and high-performance Java Web applications using J2EE technology: Using J2EE Technology. Addison-Wesley, Reading (2002)
6. Stevens, W.R.: Advanced Programming in the UNIX Environment. Addison-Wesley Professional, Reading (1992)
7. Dollimore, J., Kindberg, T., Coulouris, G.: Distributed Systems: Concepts and Design, 4th edn. Addison-Wesley, Reading (2005)
8. Silberschatz, A., Galvin, P.B., Gagne, G.: Operating System Concepts, 7th edn. John Wiley & Sons, Chichester (2004)
9. The Internet Society: Hypertext transfer protocol – http/1.0. Web (May 1996), http://tools.ietf.org/html/rfc1945
10. Gourley, D., Totty, B., Sayer, M., Reddy, S., Aggarwal, A.: HTTP: The Definitive Guide. O'Reilly, Sebastopol (2002)
11. Hasan, J., Tu, K.: Performance Tuning and Optimizing ASP .NET Applications. Apress (2003)
12. Urgaonkar, B., Pacifici, G., Shenoy, P., Spreitzer, M., Tantawi, A.: An analytical model for multi-tier internet services and its applications. In: SIGMETRICS 2005: Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems, pp. 291–302. ACM Press, New York (2005)
13. Haines, S.: Pro Java EE 5 Performance Management and Optimization. Apress (2006)

[12] See http://www.erlang.org/

14. Beloglavec, S., Heričko, M., Jurič, M.B., Rozman, I.: Analysis of the limitations of multiple client handling in a java server environment. SIGPLAN Not. 40(4), 20–28 (2005)
15. Li, P., Zdancewic, S.: Combining events and threads for scalable network services implementation and evaluation of monadic, application-level concurrency primitives. SIGPLAN Not. 42(6), 189–199 (2007)
16. Wilkins, G.: Jetty 6 architecture. Web (November 2006), http://docs.codehaus.org/display/JETTY/Architecture
17. Queinnec, C.: Inverting back the inversion of control or, continuations versus page-centric programming. SIGPLAN Not. 38(2), 57–64 (2003)
18. Gomez, J.C., Ramos, J.R., Rego, V.: Signals, timers, and continuations for multithreaded user-level protocols. Softw. Pract. Exper. 36(5), 449–471 (2006)
19. W3C: The xmlhttprequest object. Web (April 2008), http://www.w3.org/TR/XMLHttpRequest/

Creating a 'Cloud Storage' Mashup for High Performance, Low Cost Content Delivery

James Broberg[1], Rajkumar Buyya[1], and Zahir Tari[2]

[1] Department of Computer Science and Software Engineering,
The University of Melbourne, Australia
[2] Department of Computer Science and Information Technology,
RMIT University, Australia

Abstract. Many 'Cloud Storage' providers have launched in the last two years, providing internet accessible data storage and delivery in several continents that is backed by rigourous Service Level Agreements (SLAs), guaranteeing specific performance and uptime targets. The facilities offered by these providers is leveraged by developers via provider-specific Web Service APIs. For content creators, these providers have emerged as a genuine alternative to dedicated Content Delivery Networks (CDNs) for global file storage and delivery, as they are significantly cheaper, have comparable performance and no ongoing contract obligations. As a result, the idea of utilising Storage Clouds as a 'poor mans' CDN is very enticing. However, many of these 'Cloud Storage' providers are merely basic storage services, and do not offer the capabilities of a fully-featured CDN such as intelligent replication, failover, load redirection and load balancing. Furthermore, they can be difficult to use for non-developers, as each service is best utilised via unique web services or programmer APIs. In this paper we describe the design, architecture, implementation and user-experience of MetaCDN, a system that integrates these 'Cloud Storage' providers into an unified CDN service that provides high performance, low cost, geographically distributed content storage and delivery for content creators, and is managed by an easy to use web portal.

1 Introduction

Content creators, ranging from large media companies to smaller, independent start-ups have a need to store and distribute large files (such as audio and video files and rich documents) cost effectively, but with both a global reach and good performance for end-users (consumers) of these files. Traditional CDNs such as Akamai [1] can be too expensive for all but the largest enterprise customers [2]. Most major CDN providers do not publish prices but are anecdotally $2-15$ times more expensive, and require $1-2$ year commitments[1]. 'Cloud storage' providers such as Amazon S3 and Nirvanix SDN are an appealing alternative, as they provide internet accessible data storage and delivery services in several continents

[1] Information obtained from http://www.cdnpricing.com, part of a popular website and blog for CDN and streaming media professionals run by StreamingMedia.com.

G. Feuerlicht and W. Lamersdorf (Eds.): ICSOC 2008, LNCS 5472, pp. 178–183, 2009.

that are backed by rigourous Service Level Agreements (SLAs), guaranteeing specific performance and uptime targets. They offer *utility* pricing (only pay for what you use), and have no ongoing commitments or obligations. As such, a content creator can choose to harness these services only when required for peak load [3].

These emerging services have reduced the cost of content storage and delivery by several orders of magnitude, but they can be difficult to use for non-developers, as each service is best utilised via unique web services or programmer API's, and have their own unique quirks. Many websites have utilised individual Storage Clouds to deliver some or all of their content [3], most notably the New York Times [4] and SmugMug [5], however there is no general purpose, reusable framework to interact with multiple Storage Cloud providers and leverage their services as a unified CDN. Furthermore, a customer may need coverage in more locations than offered by a single provider. To address these issues, in Section 2 we introduce MetaCDN, a system which utilises numerous storage providers in order to create an overlay network that can be used as a high performance, reliable and redundant geographically distributed CDN. MetaCDN makes it easy to harness the performance and coverage footprint of multiple 'Cloud Storage' providers, removes single vendor lock-in, and allows content creators to agilely manage their deployments by expanding and contracting them as needed.

In this paper we focus on the design, architecture, implementation and user-experience of the MetaCDN system. Interested readers are directed to our previous work [6] which gives an extensive background on the 'Cloud Storage' providers used by MetaCDN (listed in Table 1), and demonstrates that they provide sufficient performance (i.e. predictable and sufficient response time and throughput) that is consistent with previous performance studies of dedicated content delivery networks [7,8].

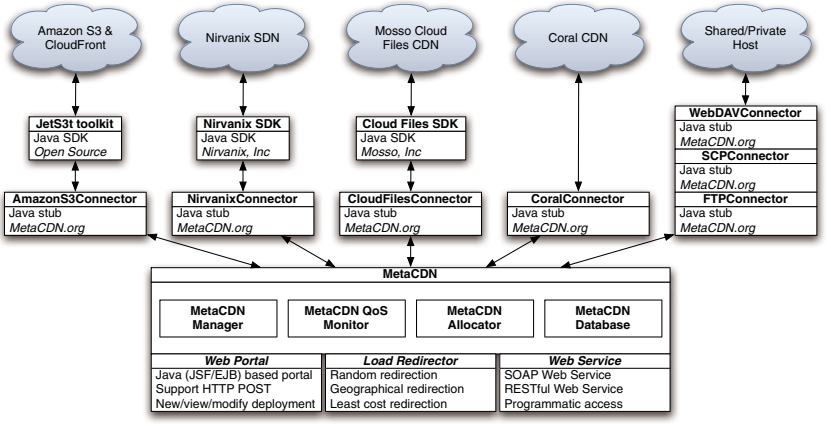


Fig. 1. MetaCDN Architecture

Table 1. Cloud Features (< 2TB for Nirvanix, < 10TB for Amazon, < 5TB for Mosso)

Feature	Nirvanix SDN	Amazon S3	Mosso Cloud Files	Coral CDN
SLA	99.9	99-99.9	*	None
Max. File Size	256GB	5GB	5GB	50MB
US PoP	Yes	Yes	Yes	Yes
EU PoP	Yes	Yes	Yes[2]	Yes
Asia PoP	Yes	Yes[3]	Yes[2]	Yes
Australasia PoP	No	No	Yes[2]	Yes
Per File ACL	Yes	Yes	Yes	No
Automatic Replication	Yes	Yes[3]	Yes[2]	Yes
Developer API	Yes	Yes	Yes	No
Bittorrent Support	No	Yes	No	No
Sideloading Support	Yes	No	No	No
Incoming data ($/GB)	0.18	0.10	0.00	0.00
Outgoing data ($/GB)	0.18	0.17(US/EU), 0.21(HK)[3], 0.22(JP)[3]	0.22	0.00
Storage ($/GB)	0.25	0.15(US), 0.18(EU)	0.15	0.00
Requests ($/1,000 PUT)	0.00	0.01	0.02	0.00
Requests ($/1,000 GET)	0.00	0.01	0.00	0.00

2 The MetaCDN System

The aim of the MetaCDN system is to to build a low cost, high performance CDN that harnesses the power of 'Storage Clouds', and is presented to users as a cohesive, unified interface. In this section we discuss the design, architecture, implementation and user-experience of the MetaCDN system.

2.1 Overall Design and Architecture of the System

The MetaCDN service (shown in Figure 1) is made available to users as a web portal, allowing users to harnessing the storage, performance capabilities and locality of multiple Cloud Storage providers, whilst hiding the complexity involved with interacting with these different entities. The web portal is most suited for small or ad-hoc deployments. A SOAP Web Service has been developed for MetaCDN that is useful for users with more complex and frequently changing content delivery needs. A RESTful Web Service is also under development, that will provide a lightweight Cloud Storage solution for mashup developers that is trivial to utilise. The web portal was developed using Java Enterprise and Java Server Faces (JSF) technologies, with a MySQL back-end to store user accounts, deployments, and the capabilities and pricing of service providers. The MetaCDN system integrates with its' upstream providers via *connectors*, which are discussed further in Section 2.2.

The web portal acts as the entry point to the system and also functions as an application-level load balancer for end-users that wish to download content

[2] When used with Limelight's CDN service.
[3] When used with Amazon's CloudFront CDN service.

that has been deployed by MetaCDN. In order to utilise the MetaCDN system effectively, content must be deployed and managed via the portal or the Web Service, as MetaCDN is unaware of content uploaded directly to participating providers. Currently, the MetaCDN portal (and backing MySQL database) is deployed at a single location (Melbourne, Australia) but in the near future we intend to deploy MetaCDN portals (and replicated backing databases) in all major continents to improve responsiveness and locality for users of MetaCDN, and consumers of the content deployed by the system. This aspect is discussed further in Section 2.5.

The MetaCDN system offers a number of functions via the web portal interface[4], including:

1. The creation of an account in the MetaCDN system, where a user registers their details, as well as credentials for any service providers (listed in Table 1) they wish to utilise.
2. Intelligent deployment of content based on geographical regions of the user's choice, their storage and transfer budget, or specific quality of service parameters (described in Section 2.3).
3. Viewing, modifying or deleting existing content deployment.
4. Viewing the physical location of deployed content replicas as a Google Maps Geolocation mashup (described in Section 2.4).

2.2 Integrating 'Cloud Storage' Providers

The MetaCDN system integrates with each storage provider via a *connector* that provides an abstraction to hide the complexity arising from each provider having their own unique Web Service API. An abstract class, *DefaultConnector*, is defined that prescribes the basic functionality that each provider could be expected to support, that must be implemented for all existing and future connectors. These include basic operations like creation, deletion and renaming of files and folders, and more advanced operations like creating Bittorrent deployments, and sideloading files (replicating a file from a publicly available *origin* URL). If an operation is not supported on a particular service, then the connector for that service should throw a *FeatureNotSupportedException*.

Whilst the intent of the MetaCDN system is to provide its users a consistent, unified interface to disparate 'Cloud Storage' systems, there are some important differences in functionality and cost between the various providers (as noted in Table 1). For example, Amazon S3 supports Bittorrent deployment of files, whilst the other providers do not. There are also differences in the largest file size that can be deployed, or whether files can be sideloaded as well as directly uploaded to a given service. MetaCDN users do not need to be aware of these subtle differences, as the content they wish to replicate is intelligently matched to the most appropriate provider that suits their specific requirements.

[4] A screencast of the web interface is available at http://www.metacdn.org

2.3 Content Deployment Options

Users of the MetaCDN web portal are presented with a number of different deployment options for replicating their content. These include:

1. Maximising coverage and performance, where MetaCDN deploys as many replicas as possible to all available locations.
2. Deploying content to specific locations a user nominates, where MetaCDN matches the requested regions with providers that service those areas.
3. Cost optimised deployment, where MetaCDN deploys as many replicas in the locations requested by the user as their storage budget will allow.
4. Quality of Service (QoS) optimised deployment, where MetaCDN deploys to providers that match specific QoS targets that a user specifies, such as average throughput or response time from a particular location, which is tracked by persistent probing from the *MetaCDN QoS monitor*.

Once a user deploys using the options above, they are either returned a set of publicly accessible URLs, pointing to the specific locations of the replica files, or a single MetaCDN URL, `http://www.metacdn.org/FileMapper.jsp?itemid=XX`, where `XX` is a unique hash key associated with the deployed content. This provides a single namespace which may be more convenient for users, and can provide automatic and totally transparent load balancing for end-users. This functionality is described further in Section 2.5.

2.4 Integration of Geo-IP Services and Google Maps

Cloud Storage offerings are already available from providers located across the globe. The principle of cloud computing and storage is that you shouldn't need to care where the processing occurs, or where your data is stored - the services are essentially a black box. However, your software and data are subject to the laws of the nations they are executed and stored in. Cloud storage users could find themselves inadvertently running afoul of the Digital Millennium Copyright Act (DMCA)[5] or Cryptography Export laws that may not apply to them in their own home nations. As such, it is important for Cloud Storage users to know precisely where their data is stored. Furthermore, this information is crucial for MetaCDN load balancing purposes, so end-users are redirected to the closest replica, to maximise their download speeds and minimise latency. To address this issue, MetaCDN offers its' users the ability to pinpoint exactly where their data is stored via geolocation services and Google Maps integration. When MetaCDN deploys replicas to different Cloud Storage providers, they each return a URL pointing to the location of the replica. MetaCDN then utilises a geolocation service (either free[6] or commercial[7]) to find the latitude and longitude of where the file is stored. This information is stored in the MetaCDN database, and can be overlaid onto a Google Maps view inside the MetaCDN portal, giving users a birds-eye view of where their data is currently being stored.

[5] Available at http://www.copyright.gov/legislation/dmca.pdf
[6] Hostip.info is a free community-based project to geolocate IP addresses.
[7] MaxMind GeoIP is a commercial IP geolocation service.

2.5 Load Balancing via DNS and HTTP Redirection

Load balancing for both MetaCDN users and consumers of the content that the system replicates is achieved in two stages. First, users or consumers of MetaCDN are directed to their closest portal at the DNS resolution stage. Currently, there is a MetaCDN portal running in Australia, and soon their will be portals running in Europe and North America. For MetaCDN consumers, if they are attempting to access a file via a MetaCDN URL, then they are redirected (by a `HTTP 302 Found` directive) to the most appropriate replica. What constitutes the most appropriate replica depends on the deployer of the content (and the preferences they expressed, described in Section 2.3). This could be the highest performing replica, the geographically closest replica or even the cheapest replica.

3 Conclusion

In this paper we gave an overview of the design, architecture, implementation and user-experience of MetaCDN, a system that integrates 'Cloud Storage' providers into an unified CDN service that provides high performance, low cost, geographically distributed content storage and delivery for content creators, and is managed by an easy to use web portal. More information on the ongoing development of MetaCDN can be found at `http://www.metacdn.org`

This work is supported by Australian Research Council (ARC) as part of the Discovery Grant 'Coordinated and Cooperative Load Sharing between Content Delivery Networks' (DP0881742, 2008-2010).

References

1. Maggs, B., Technologies, A.: Global internet content delivery. In: First IEEE/ACM International Symposium on Cluster Computing and the Grid, pp. 12–12 (2001)
2. Pathan, M., Buyya, R.: A Taxonomy of CDNs. Content Delivery Networks, 33–78 (2008)
3. Elson, J., Howell, J.: Handling Flash Crowds from your Garage. In: USENIX 2008: 2008 USENIX Annual Technical Conference (June 2008)
4. Gottfrid, D.: Self-service, prorated super computing fun! OPEN: All the code that is fit to print (2007), `http://open.nytimes.com/2007/11/01/self-service-prorated-super-computing-fun`
5. MacAskill, D.: Scalability: Set Amazon's Servers on Fire, Not Yours. In: ETech 2007: O'Reilly Emerging Technology Conference (2007), `http://blogs.smugmug.com/don/files/ETech-SmugMug-Amazon-2007.pdf`
6. Broberg, J., Buyya, R., Tari, Z.: MetaCDN: Harnessing 'Storage Clouds' for high performance content delivery. Technical Report GRIDS-TR-2008-10, Grid Computing and Distributed Systems Laboratory, The University of Melbourne (August 2008)
7. Johnson, K., Carr, J., Day, M., Kaashoek, M.: The measured performance of content distribution networks. Computer Communications 24(2), 202–206 (2001)
8. Su, A., Choffnes, D., Kuzmanovic, A., Bustamante, F.: Drafting behind Akamai (travelocity-based detouring). ACM SIGCOMM Computer Communication Review 36(4), 435–446 (2006)

First International Workshop on Quality-of-Service Concerns in Service Oriented Architectures (QoSCSOA 2008)

Introduction: First International Workshop on Quality of Service Concerns in Service Oriented Architectures (QoSCSOA 2008)

Liam O'Brien and Paul Brebner

NICTA, Building A, 7 London Circuit, Canberra, ACT 2601, Australia
RSISE, Australian National University, Canberra, ACT 0200, Australia
{Liam.OBrien,Paul.Brebner}@nicta.com.au

Keywords: Quality-of-Service, Quality Attributes, Service Oriented Architecture.

1 Workshop Description

Service-Oriented Architecture (SOA) is having a substantial impact on the way software systems are developed. Today many systems are being designed and developed that use an SOA style. Although some progress has been made on several fronts on addressing Quailty-of-Service (QoS) concerns in SOAs much research is still needed in addressing QoS issues in the design, development and operation of SOA-based systems. This workshop focuses on techniques and approaches for managing QoS concerns throughout the entire lifecycle of SOA-based systems. The following topics are of interest in this workshop:

- Techniques for determining quality requirements for SOA-based systems
- Techniques, patterns and approaches for handling specific quality attribute requirements in the design of SOA-based systems
- Techniques, patterns and approaches for handling specific quality attribute requirements in the implementation of SOA-based systems
- Deployment and Monitoring of SOA-based systems
- Resourcing models to guarantee specific QoS requirements (virtualisation, grid, etc)
- QoS aspects of virtualised SOA-based systems
- Assessment techniques and approaches for specific qualities of SOA-based systems including modeling and simulation of specific qualities
- Service Level Agreements (SLAs) in an SOA context including their development and negotiation
- Validation of properties/service qualities in SOA-based systems
- Economics of handling specific QoS requirements in SOA-based systems
- Managing QoS concerns for SOA-based systems throughout the entire software life cycle
- Autonomic QoS management in SOA-based systems
- Relationship of QoS of SOA-based systems to the underlying business processes

G. Feuerlicht and W. Lamersdorf (Eds.): ICSOC 2008, LNCS 5472, pp. 187–188, 2009.

2 Workshop Objectives

The objective of this workshop is to bring together researchers and practitioners with experience in QoS issues in SOAs. The workshop aims to determine the current state of the practice in determining, using and managing QoS and non-functional requirements throughout the entire life cycle of an SOA-based system. It is proposed to determine the current state of the art in this area and outline a roadmap of potential research directions.

3 Motivation

As SOA technology matures and organisations are adopting and building business and mission critical systems using the SOA approach there is a growing need to make sure that such systems meet their non-functional and QoS requirements as well as their functional requirements. It is important that the non-functional and QoS requirements of such SOA-based systems are determined early in the life cycle and the systems are designed, built, and deployed in such a way so as to meet these requirements. The aim of this workshop is to examine how non-functional and QoS requirements are captured, used and managed throughout the entire life cycle of SOA-based systems.

4 Workshop Format

The half-day workshop consists of a keynote by Adrian Mos on the theme of "*Challenges in Integrating Tooling and Monitoring for QoS Provisioning in SOA Systems*".

The following technical papers were accepted for presentation at the workshop:

- *Challenges for Assuring Quality of Service in a Service-Oriented Environment,* Sriram Balasubramaniam, Grace A. Lewis, Ed Morris, Soumya Simanta, and Dennis B. Smith (*paper not included in proceedings*)
- *A Scalable Approach for QoS-based Web Service Selection,* Mohammad Alrifai, Thomas Risse, Peter Dolog, and Wolfgang Nejdl
- *Towards QoS-based Web Services Discovery,* Jun Yan and Jingtai Piao
- *A Redundancy Protocol for Service-Oriented Architectures,* Nicholas May
- *A Context-aware Trust Model for Service-oriented Multi-agent Systems,* Kaiyu Wan and Vasu Alagar

The workshop also includes a discussion session led by Vladimir Tosic on the topic *"Three Common Mistakes in Modeling and Analysis of QoS of Service-Oriented Systems"*.

Challenges in Integrating Tooling and Monitoring for QoS Provisioning in SOA Systems (Keynote)

Adrian Mos

INRIA, 655 avenue de l'Europe, Montbonnot 38334, France
Adrian.Mos@inria.fr

Abstract. When addressing QoS concerns in SOA systems, some of the important challenges lie in the seamless integration of user-side tooling support with the actual execution of services and business processes. The user must be able to easily express QoS constraints and associate them with service definitions, compositions and orchestrations. The constraints must be taken into account at all levels in the increasingly complex SOA infrastructures and used to monitor and enforce SLAs, as well as to potentially drive adaptation mechanisms to improve end-to-end QoS. Conversely, QoS data generated by the execution of services and processes must be made available to the user in ways that ensure that appropriate information is presented at different levels of abstraction with the right level of granularity in order to improve understanding and future planning of service-based functionality.

These challenges can be grouped in two main areas: *Driving Monitoring and Adaptation from SOA Tooling* and *Integrating Dynamic Monitoring Information in SOA Tooling*. The first area includes means of providing simple and expressive QoS specifications within a variety of tooling options, including heavyweight editors and lightweight Web 2.0 based designers. It also includes approaches to generate appropriate monitoring artefacts (probe-specific information for provenance determination) and adaptation rules and options based on expressed QoS requirements.

The second area includes challenges in obtaining proper monitoring information at all relevant SOA levels (Enterprise Service Buses, component-platforms such as SCA, process engines as well as lower-level grid infrastructures). It equally relates to transporting and presenting it to a large variety of editors and SOA-related visual tools (for architecture definition, process specification or service mash-ups in both heavyweight and lightweight user environments). Apart from presenting useful information to the user, QoS information must also be stored and updated in service repositories for later user-referral, as well as for use in self-adaptive environments where processes and compositions can dynamically change in order to improve QoS.

This presentation gives an overview of these challenges and presents ongoing work undertaken by INRIA for addressing them in the context of different research projects as well as the internal INRIA initiative called "*galaxy*". Such work includes a metamodel-based approach to transformations called the STP-Intermediate Model which is part of the Eclipse SOA Tools Platform Project and that aims at unifying the SOA tooling space and relating it to monitoring data.

Keywords: SOA, quality of service (QoS), monitoring, modelling, tooling.

G. Feuerlicht and W. Lamersdorf (Eds.): ICSOC 2008, LNCS 5472, p. 189, 2009.

A Scalable Approach for QoS-Based Web Service Selection

Mohammad Alrifai[1], Thomas Risse[1], Peter Dolog[2], and Wolfgang Nejdl[1]

[1] L3S Research Center
Leibniz University of Hannover, Germany
{alrifai,risse,nejdl}@L3S.de
[2] Department of Computer Science
Aalborg University, Denmark
dolog@cs.aau.dk

Abstract. QoS-based service selection aims at finding the best component services that satisfy the end-to-end quality requirements. The problem can be modeled as a multi-dimension multi-choice 0-1 knapsack problem, which is known as NP-hard. Recently published solutions propose using linear programming techniques to solve the problem. However, the poor scalability of linear program solving methods restricts their applicability to small-size problems and renders them inappropriate for dynamic applications with run-time requirements. In this paper, we address this problem and propose a scalable QoS computation approach based on a heuristic algorithm, which decomposes the optimization problem into small sub-problems that can be solved more efficiently than the original problem. Experimental evaluations show that near-to-optimal solutions can be found using our algorithm much faster than using linear programming methods.

1 Introduction

Industrial practice witnesses a growing interest in the ad-hoc model for service composition in the areas of supply chain management, accounting, finances, eScience as well as in multimedia applications. With the growing number of available services the composition problem becomes a decision problem on the selection of component services from a set of alternative services that provide the same functionality but differ in quality parameters.

Given an abstract representation of a composition request (e.g. in a workflow language like BPEL [1]), and given a list of functionally-equivalent web service candidates for each task in the composition request, the goal of service selection algorithms is to find one web service from each list such that the overall QoS is optimized and user's end-to-end QoS requirements are satisfied. This problem can be modeled as *Multi-Choice Multidimensional Knapsack problem* (MMKP), which is known to be NP-hard in the strong sense [2]. Therefore it can be expected that any exact solution to MMKP has an exponential cost. In the dynamic environment of web services, where deviations from the QoS estimates

G. Feuerlicht and W. Lamersdorf (Eds.): ICSOC 2008, LNCS 5472, pp. 190–199, 2009.

occur and decisions upon replacing some services has to be taken at run-time (e.g. in multimedia applications), the efficiency of the applied service selection algorithm becomes crucial.

Due to the poor scalability of *(Mixed) Integer Linear programming* (MILP) methods [3], recently proposed solutions like [4,5] fail short in addressing run-time requirements. In this paper we propose an efficient and scalable heuristic approach for the QoS-based service selection problem. The contribution of this paper can be stated as follows:

- We map the global QoS optimization problem into sub-problems that can be solved more efficiently by local QoS optimization.
- We show how the results of the problem decomposition can be applied in a distributed architecture that includes a service composer and a set of distributed service brokers.

Our heuristic approach to QoS-based service selection does not necessarily result in "the" optimal set of services. Nevertheless, since the business requirements (such as response times or throughput) are only approximate, we need to find a reasonable set of services that covers the requirements approximately at acceptable costs and avoids obvious violations of constraints. The experimental evaluation we present in this paper show that our approach outperforms all previous solutions in terms of computational complexity but still gives qualitative comparable results.

The rest of the paper is organized as follows. In the next section we discuss related solutions. Section 3 introduces the system model and gives a problem statement. Our approach for a scalable QoS computation for web service selection is presented in section 4. In section 5 we discuss the results from our experimental evaluation. Finally, section 6 gives conclusions and an outlook on possible continuations of our work.

2 Related Work

Recently, the QoS-based web service selection and composition in service-oriented applications has gained the attention of many researchers [4,6,5,7]. In [6] the authors propose an extensible QoS computation model that supports open and fair management of QoS data. The problem of QoS-based composition is not addressed by this work. The work of Zeng at al. [4] focuses on dynamic and quality-driven selection of services. The authors use global planning to find the best service components for the composition. They use (mixed) linear programming techniques [3] to find the optimal selection of component services. Similar to this approach Ardagna et al. [5] extends the linear programming model to include local constraints. Linear programming methods are very effective when the size of the problem is small. However, these methods suffer from poor scalability due to the exponential time complexity of the applied search algorithms [8]. In [7] the authors propose heuristic algorithms to find a near-to-optimal solution more efficiently than exact solutions. The time complexity of the heuristic algorithm (WS_HEU) is polynomial. Despite

the significant improvement of this algorithm compared to exact solutions, it dose not scale with respect to an increasing number of web services and remain out of the real-time requirements.

3 System Model and Problem Statement

In our model we assume that we have a universe of web services $\mathbb{S}$ which is defined as a union of *abstract service classes*. Each abstract service class $S_j \in \mathbb{S}$ is used to describe a set of functionally-equivalent web services (e.g. Lufthansa and Qantas flight booking web services). In this paper we assume that information about service classes is managed by a set of service brokers as described in [6,9]. Web services can join and leave service classes at any time by means of a subscription mechanism.

3.1 Abstract vs. Concrete Composite Services

We also distinguish between the following two concepts:

- An abstract composite service, which can be defined as an abstract representation of a composition request $CS_{abstract} = \{S_1, \ldots, S_n\}$. $CS_{abstract}$ refers to the required service classes without referring to any concrete web service.
- A concrete composite service, which can be defined as an instantiation of an abstract composite service. This can be obtained by binding each abstract service class (e.g. flight booking) in $CS_{abstract}$ to a concrete web service s_j, $s_j \in S_j$ (e.g. Qantas flight booking web Service).

3.2 QoS Criteria

In our study we consider quantitative non-functional properties of web services, which can be used to describe the quality of a service s. We use the vector $Q_s = \{q_1, q_2, \ldots, q_r\}$ to represent these properties. These can include generic QoS attributes like response time, availability, price, reputation etc, as well as domain-specific QoS attributes like bandwidth, video quality for multimedia web services. The values of these QoS attributes can be either collected from service providers directly (e.g. price), recorded from previous execution monitoring (e.g. response time) or from user feedbacks (e.g. reputation) [6]. The set of QoS attributes can be divided into two subsets: positive and negative attributes. The values of positive attributes need to be maximized (e.g. availability), whereas the values of negative attributes need to be minimized (e.g. response time). For the sake of simplicity, we consider only negative attributes (positive attributes can be easily transformed into negative attributes by multiplying their values by -1). We use the function $q_i(s)$ to determine the i-th quality parameter of service s.

3.3 QoS Computation of Composite Services

The QoS value of a composite service is decided by the QoS values of its component services as well as the composition model used (e.g. sequential, parallel,

conditional and/or loops). In this paper, we focus on the service selection algorithm for QoS-based service composition, and its performance on the sequential composition model. Other models may be reduced or transformed to the sequential model. Techniques for handling multiple execution paths and unfolding loops from [4], can be used for this purpose.

The QoS vector for a composite service CS is defined as $Q_{CS} = \{q'_1(CS), \ldots, q'_r(CS)\}$ where $q'_i(CS)$ represents the estimated QoS values of a composite service CS and can be aggregated from the expected QoS values of its component services. Table 1 shows examples of some QoS aggregation functions.

Similar to [4,6,5,7], we assume in our model that QoS aggregation functions can be linearized and represented by the summation relation. For QoS attributes that are typically aggregated as a product (e.g. availability) we apply a logarithm operation to transform them into a summation relation. We extend our model to support the following aggregation function:

$$q'_k(CS) = \sum_{j=1}^{n} q_k(s_j) \tag{1}$$

Table 1. Examples of QoS aggregation functions for composite services

QoS Attribute	Aggregation Function
Response Time	$q'_{res}(CS) = \sum_{j=1}^{n} q_{res}(s_j)$
Price	$q'_{price}(CS) = \sum_{j=1}^{n} q_{price}(s_j)$
Availability	$q'_{av}(CS) = \prod_{j=1}^{n} q_{av}(s_j)$

3.4 Utility Function

In order to evaluate the multi-dimensional quality of a given web service composition a utility function is used. In this paper we use a Multiple Attribute Decision Making approach for the utility function: i.e. the *Simple Additive Weighting (SAW)* technique [10]. The utility computation involves scaling the values of QoS attributes to allow a uniform measurement of the multi-dimensional service qualities independent of their units and ranges. The scaling process is then followed by a weighting process for representing user priorities and preferences. In the scaling process each QoS attribute value is transformed into a value between 0 and 1, by comparing it with the minimum and maximum possible aggregated value. These values can be easily estimated by aggregating the local minimum (or maximum) possible value of each service class in CS. For example, the maximum execution price of any concrete composite service can be computed by summing up the execution price of the most expensive service in each service class. Formally, we compute the minimum and maximum aggregated value of the k-th QoS attribute as follows:

$$Qmin'(k) = \sum_{j=1}^{n} Qmin(j,k) \quad \text{and} \quad Qmax'(k) = \sum_{j=1}^{n} Qmax(j,k) \tag{2}$$

where $Qmin(j,k) = \min_{\forall s_{ji} \in S_j} q_k(s_{ji})$ is the minimum value (e.g. minimum price) and $Qmax = \max_{\forall s_{ji} \in S_j} q_k(s_{ji})$ is the maximum value (e.g. maximum price) that can be expected for service class S_j according to the available information about service candidates of this class.

Now the overall utility of a composite service is computed as

$$U'(CS) = \sum_{k=1}^{r} \frac{Qmax'(k) - q'_k(CS)}{Qmax'(k) - Qmin'(k)} \cdot w_k \tag{3}$$

with $w_k \in \mathbb{R}_0^+$ and $\sum_{k=1}^{r} w_k = 1$ being the weight of q'_k to represent user's priorities.

The utility function $U'(CS)$ is used to evaluate a given set of alternative service compositions. However, finding the best composition requires enumerating all possible combinations of service candidates. For a composition request with n service classes and l service candidate per class, there are l^n possible combinations to be examined. Performing exhaustive search can be very expensive in terms of computation time and, therefore, inappropriate for applications with many services and dynamic needs.

3.5 Problem Statement

The problem of finding the best service composition without enumerating all possible combinations is considered as an optimization problem, in which the overall utility value has to be maximized while satisfying all global constraints. Formally, the optimization problem we are addressing can be stated as follows:

For a given abstract composite service $CS_{abstract} = \{S_1, \ldots, S_n\}$ with a set of m global QoS constraints $C' = \{c'_1, \ldots, c'_m\}$, find an implementation $CS = \{s_{1b}, \ldots, s_{nb}\}$ by bounding each S_j to a concrete service $s_{jb} \in S_j$ such that:

1. The overall utility $U'(CS)$ is maximized, and
2. The aggregated QoS values satisfy: $q'_k(CS) \leq c'_k, \forall c'_k \in C'$

4 A Scalable QoS Computation

In this section we present a scalable solution to the QoS-bases web service composition problem. We decompose the global optimization problem into subproblems that can be solved independently. For this purpose, we first map the global QoS computation on the composite service level $U'(CS)$ into local computations that can be performed on each service class independently (see Section 4.1). Second, we propose a simple algorithm for decomposing each global QoS constraint $c'_k \in C'$ into n local constraints that can be verified locally on the component services (see Section 4.2). Finally, we present a distributed service selection algorithm that leverages local search for achieving global QoS requirments (see Section 4.3)

4.1 Decomposition of Global QoS Computation

The use of (3) on the composite service level requires enumerating all possible combinations of the service candidates to find the optimal selection. This approach can be very inefficient for large scale problems or applications with run-time requirements. Therefore, we derive a modified utility function $U'_{local}(s)$ from $U'(CS)$ that can be applied on the component service level, without the need for evaluating all possible combinations.

By applying (1) and (2) we get:

$$\begin{aligned} U'(CS) &= \sum_{k=1}^{r} \frac{\sum_{j=1}^{n} Qmin(j,k) - \sum_{j=1}^{n} q_k(s_j)}{Qmax'(k) - Qmin'(k)} \cdot w_k \\ &= \sum_{j=1}^{n} \underbrace{\left(\sum_{k=1}^{r} \frac{Qmax(j,k) - q_k(s_j)}{Qmax'(k) - Qmin'(k)} \cdot w_k\right)}_{U'_{local}(s)} = \sum_{j=1}^{n} U'_{local}(s_j) \end{aligned} \tag{4}$$

The utility function $U'_{local}(s)$ can be computed for each service class S_j independently, provided that the global parameters $Qmin'$ and $Qmax'$ are specified. These parameters can be easily computed beforehand by aggregating the local maximum and minimum values of each service class. Thus, by selecting the service candidate with the maximum U'_{local} value from each class, we can ensure that $U'(CS)$ is maximized (i.e. satisfying the first requirement in 3.5).

4.2 Decomposition of Global Constraints

To ensure that the outcome of the local QoS computation satisfies global QoS constraints (i.e. the second requirement in 3.5), we need to decompose each global constraint $c'_k \in C', 1 \leq k \leq m$ into n local constraints. We use local statistics about the quality values to estimate a reasonable decomposition of each global constraint c'_k as follows:

1. Initially set the local constraint value c_{jk} of each service class to the local maximum value of that class,

$$\forall c'_k \in C' : c_{kj} = Qmax(j,k), 1 \leq j \leq n \tag{5}$$

2. Compute the difference between the global constraint values and the aggregated value of the local constraints,

$$\forall c'_k \in C' : d_k = \sum_{j=1}^{n} c_{kj} - c'_k \tag{6}$$

3. Adjust the current set of local constraints based on the relative distance between the local maximum and minimum QoS value using the following formula:

$$\forall c'_k \in C' : c_{kj} = c_{kj} - d_k * \frac{Qmax(j,k) - Qmin(j,k)}{\sum_{x=1}^{n}(Qmax(x,k) - Qmin(x,k))}, 1 \leq j \leq n \tag{7}$$

4.3 Distributed Optimization of the QoS Computation

We assume an architecture consisting of a *service composer* and a number of *service brokers* - either distributed or on a single machine. Each service broker is responsible for managing QoS information of a set of web service classes. A list of available web services is maintained by the service broker along with registered measurements of their non-functional properties, i.e. QoS attributes, like response time, throughput, price etc. The service composer instantiates a composite service CS in interaction with the service brokers.

The procedure of the distributed QoS-based service composition is depicted in figure 1. The service composer requests statistical information for each service class from the responsible service brokers, namely, $Qmax(j,k)$ and $Qmin(j,k)$ and computes the global parameters: $Qmax'(k)$, $Qmin'(k)$, $1 \leq k \leq r$. Each global constraint $c'_k, 1 \leq k \leq m$ is decomposed as described in section 4.2 into a set of local constraints $c_{1k}, ..., c_{jk}$. The composer sends these local constraints along with the global parameters $Qmax'(k)$, $Qmin'(k)$, $1 \leq k \leq r$ to each service broker. Each service broker performs a local search and returns the best service candidate that satisfies the local constraints and has the maximum U'_{local} value. The service composer collects the results from the brokers and checks them for further optimization. Further optimization is possible if the total saving in the value of any of quality attributes is greater than zero. The total saving δ_k of the k-th QoS attribute is computed as:

$$\delta_k = \sum_{j=1}^{n} (c_{jk} - q_{jk}), 1 \leq k \leq m \tag{8}$$

The service composer uses the total saving in the quality value to relax the current local constraints as follows:

$$\forall c_{kj} \in C' : c_{kj} = q_{kj} + \delta_k * \frac{Qmax(j,k) - q_{jk}}{\sum_{x=1}^{n} (Qmax(x,k) - q_{xk})}, 1 \leq j \leq n \tag{9}$$

The relaxed local constraints are sent back to the service brokers for improving their local results. The procedure is repeated as long as a new solution is found. Otherwise, the procedure stops and the final composition is constructed from the currently select component services $CS = \{s_1, \ldots, s_n\}$.

Unlike the heuristic algorithm WS_HEU [7], the expected number of iterations in our algorithm is very low as we only consider upgrading the current solution by means of relaxing the constraints and no downgrading is required. By only relaxing the local constraints it is guaranteed that the new solution, if any exists, has a higher overall utility value than the current solution. Our algorithm is guaranteed to converge as after each round the service composer checks the new solution as well as the new total saving value δ_k against those of the previous round. The composer stops the optimization process as soon as there is no improvement in the utility value nor saving in the quality values.

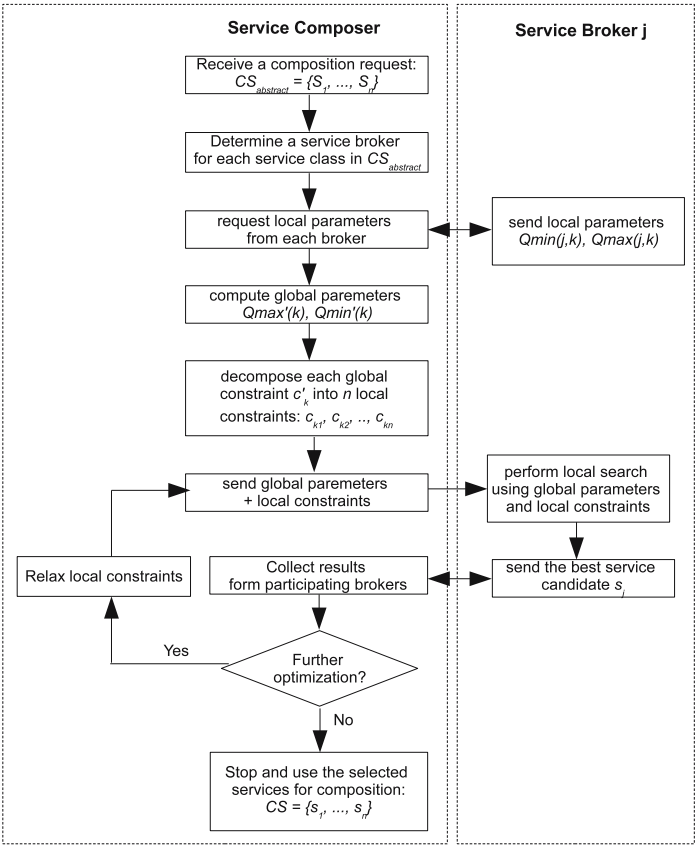


Fig. 1. Service Composer - Service Broker interactions

5 Experimental Evaluation

We have evaluated our proposed solution by means of several experiments, which we describe in this section. We conducted the experiments on a HP ProLiant DL380 G3 machine with 2 Intel Xeon 2.80GHz processors and 6 GB RAM. The machine is running under Linux (CentOS release 5) and Java 1.6.

We compare the performance and the quality of the results of our solution with those of the linear programming methods (LP) [4,5] and the heuristic algorithm WS_HEU [7]. For LP we use the open source Linear Programming system *lpsolve* version 5.5 [11]. For WS_HEU we use our own implementation. We implemented it as fair as possible taking into account all possible optimizations to reduce the computation time as much as possible. We experimented with several instances of the QoS composition problem by varying the number of service classes n and the number of service candidates per class l. Each unique combination of these parameters represents one instance of the composition problem. For the QoS data we use the QWS real dataset from [12]. This dataset includes measurements of 9 QoS attributes for 364 real web services. To evaluate the scalability of our solution,

however, we need to run experiments with a much larger set of services. Therefore, we duplicated the QWS dataset several times, each time multiplying the quality values of web services by a uniformly distributed random value between 0.1 and 2.0. In this way we achieved a data set of about 100.000 services.

5.1 Performance Evaluation

We evaluate the performance of the three approaches, by measuring the required time for finding the solution (i.e. the best combination of concrete services) by each approach. Figure 2 shows a comparison of the performance of LP, WS_HEU and our solution, which we label with DIST_HEU. In this experiment we study the performance of the three approaches with respect to the size of the problem in terms of the number of service classes n and the number of service candidates per class l. The results in both graphs (varying number of classes and varying number of candidates) show that DIST_HEU has a much better scalability than LP and WS_HEU in all problem instances (always less than 100 msec).

5.2 Optimality Evaluation

To evaluate the quality of the results of our approach, we measure the closeness of the returned results to the optimal results obtained by the LP method by calculating the optimality ration $R = \frac{U_{approx}}{U_{opt}}$. U_{approx} is the utility of the best composition returned by our approach according to (3) and U_{opt} is the utility of

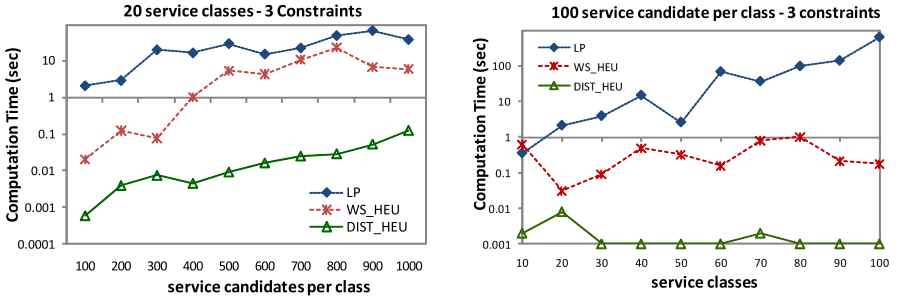


Fig. 2. Computational time with respect to the problem size

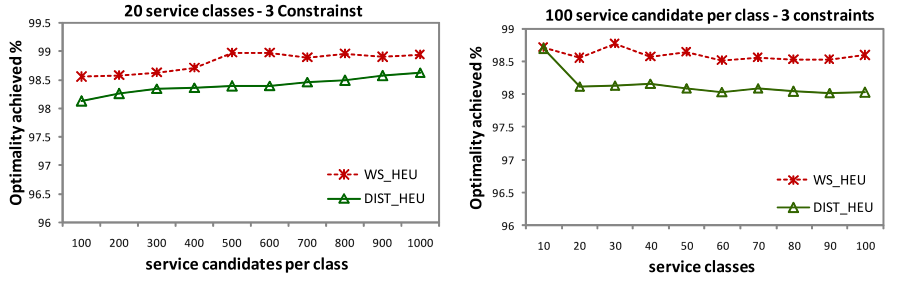


Fig. 3. Comparison of the achieved optimality with respect to the size of the problem

the composition returned by the LP method. The results shown in figure 3 indicate that DIST_HEU achieves good results with 98% optimality ratio in average. It can also be seen that result quality of our approach DIST_HEU is in average just 1% below the WS_HEU results. However, the cost of WS_HEU for this little improvement in terms of computation time is very high as we see from figure 2.

6 Conclusion and Future Work

This paper describes a scalable method for the QoS-based service selection. The problem is known to be NP-hard. Therefore heuristic solutions are commonly used. Our proposed method allows to dramatically reduce the efforts compared to existing heuristic solutions by a factor of 10 to 100 dependent on the complexity of the service environment. We decompose the global optimization problem into a number of sub-problems that can easily be solved by local optimization within each service class. In addition the decomposition allows distributing the processing to better fit to the distributed web service environment. We are currently working on extending our approach to support more complex composition models than the sequential model. Furthermore, we plan to develop a more sophisticated QoS constraint decomposition algorithm to improve the optimality of the obtained results.

References

1. OASIS: Web services business process execution language (April 2007), http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf
2. Pisinger, D.: Algorithms for Knapsack Problems. PhD thesis, University of Copenhagen, Dept. of Computer Science (1995)
3. Nemhauser, G.L., Wolsey, L.A.: Integer and Combinatorial Optimization. Wiley-Interscience, New York (1988)
4. Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., Sheng, Q.Z.: Quality driven web services composition. In: WWW, pp. 411–421 (2003)
5. Ardagna, D., Pernici, B.: Adaptive service composition in flexible processes. IEEE Trans. Software Eng. 33(6), 369–384 (2007)
6. Liu, Y., Ngu, A.H.H., Zeng, L.: Qos computation and policing in dynamic web service selection. In: WWW, pp. 66–73 (2004)
7. Yu, T., Zhang, Y., Lin, K.J.: Efficient algorithms for web services selection with end-to-end qos constraints. ACM Trans. Web 1(1), 6 (2007)
8. Maros, I.: Computational Techniques of the Simplex Method. Springer, Heidelberg (2003)
9. Li, F., Yang, F., Shuang, K., Su, S.: Q-peer: A decentralized QOS registry architecture for web services. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749, pp. 145–156. Springer, Heidelberg (2007)
10. Yoon, K.P., Hwang, C.L.: Multiple Attribute Decision Making: An Introduction (Quantitative Applications in the Social Sciences). Sage Publications, Thousand Oaks (1995)
11. Berkelaar, M., Kjell Eikland, P.N.: Open source (mixed-integer) linear programming system. Sourceforge, http://lpsolve.sourceforge.net/
12. Al-Masri, E., Mahmoud, Q.H.: Investigating web services on the world wide web. In: WWW (2008)

Towards QoS-Based Web Services Discovery

Jun Yan and Jingtai Piao

School of Information Systems and Technology
University of Wollongong
Wollongong, NSW, Australia 2522
{jyan,jp928}@uow.edu.au

Abstract. The current UDDI-based web services discovery technologies are designed to discover services which can satisfy consumers' functional requirements. The consumers' non-functional requirements such as quality of services are largely ignored in discovery. This may lead to the problem that the services returned from discovery are ineffective and even useless. To solve this problem, this paper presents an approach to achieving QoS-based Web services discovery. Data structures are proposed for both service providers and service consumers to describe non-functional information about services. A serial of algorithms are developed for matching and ranking services according to consumers' non-functional requirements.

1 Introduction

Web services is an emerging technology which provides a machine to machine interaction over a network by using a series of standardized technologies, including WSDL, SOAP and UDDI [1]. Of these technologies, UDDI is designed to be interrogated by SOAP messages, as well as to provide an access to WSDL documents which describe the protocol bindings and messages [3]. With UDDI in place, the advertisement, discovery, and binding of web services through the Internet can be achieved. IBM, SAP and Microsoft established a publicly accessible Universal Business Registry (UBR) and this is recognized as a trial of the UDDI specification. However, the adoption rate of UDDI is still remained at a low level [1]. One of the significant reasons for this stagnation is that the existing Web services discovery framework only supports discovery of services which functionally match with the user request. Discovery with consideration of services' non-functional features has been largely ignored [1].

Services' non-functional features can be best represented with the Quality of Service (QoS) offered by the web services, such as price, response time, reliability, and so on [4]. In the real world, there may be a set of web services which offer the same functional features but current UDDI cannot distinguish these functionally equivalent services. QoS should be a key factor to differentiate the services with similar functionalities [9] [6]. When a service consumer queries for such functional features, only those services that also satisfy the consumer's non-functional requirements are of interests to this consumer [2]. Service discovery based on functional matching only

G. Feuerlicht and W. Lamersdorf (Eds.): ICSOC 2008, LNCS 5472, pp. 200–210, 2009.

can lead to severe problems. For example, those services with much longer-than-expected response time or much lower-than-expected availability may be presented to a service consumer as results of service discovery. Subsequently, the performance of the consumer application can be largely deteriorated due to the use of these services. For this reason, QoS-based service discovery which aims at discovering services which meet both functional and non-functional requirements of service consumers becomes critical to the wide adoption of Web services in practice.

To address the above problem, this paper presents an approach towards QoS-based Web services discovery through the extension of the current UDDI service discovery framework. This research primarily focuses on two areas. First, this paper proposes a generic model to represent QoS information in service advertisements as well as QoS requirements in service discovery requests. Based on this model, a matching algorithm and a ranking algorithm are presented. Given a service discovery request, the matching algorithm compares its QoS requirements with the QoS advertisements in the repository and locates those services with matching QoS. The ranking algorithm furthers ranks these discovered services to facilitate the consumers to select services.

The remainder of this paper is organized as follows. Section 2 introduces major related work. Then, Section 3 presents a XML-based model to represent the QoS information, and an extension to UDDI tModel to store QoS information. In Section 4, QoS matching and ranking algorithms for service discovery are discussed based on the proposed QoS model. Finally, Section 5 concludes this paper and outlines our future work.

2 Related Work

Extension of UDDI with consideration of QoS has been recognized as a key solution of distinguishing the web service discovery results with same functionality [1][15] [20]. The first step to achieving this is the active development of QoS information representation approaches. To name a few, Mani et al [19] claimed that the major QoS requirements are performance, reliability, security, etc. Ran [1] categorized the QoS information as three domains, namely run-time related QoS, transaction related QoS and security related QoS. Dobson et al [8] proposed a reusable generic QoS ontology by categorizing the QoS information as measurable and immeasurable. Maximilien et al [7] intended to develop a comprehensive QoS ontology including the description of the QoS relationship. Moreover, Zhou et al [6] provided a novel DAML-QoS to establish a three-layer QoS ontology, including profile layer, property layer and metrics layer. Bianchini et al [10] proposed an approach which allows users to submit their QoS requirements semantically, using an ontology which categorizes the QoS attributes as numeric, Boolean and enumeration. Although these representation approaches are successful in capturing QoS information from different perspectives, the flexibility of syntactic QoS model is still lack and the semantic QoS model is relatively complex and time consuming.

Embedding QoS information within UDDI is another important research issue. Adam Blum and Fred Cater [11] stated four different QoS information sorting methods by extend tModel and bindingTemplate within UDDI. D'Melloet et al [16] introduced a QoS repository as a QoS broker to store and interact with QoS information. Similarly, Yu and Lin [17] proposed a QoS capable web service (QCWS) architecture

which is operated as a QoS broker to manage the interaction of QoS information between client and service provider. Furthermore, QCWS is designed have the capability to allocated resource to the clients.

QoS matching and ranking is the key in QoS-based service discovery. The existing work covers both syntactic matching and semantic matching. Yang and Huangcan [12] treated the QoS model as an N dimensional matrix. The priority of QoS can be described as the distance from the spot of the requirement to the provided spot in the QoS space. Wenli [15] suggested utilizing fuzzy logic to calculate a threshold value of QoS information. The services which are out of this threshold will not be discovered. Kritikos and Plexousakis [14] proposed a matching algorithm, which transfers the QoS information into Constraint Satisfaction Problem (CSP) and the matching progress is performed based on the consistency of the CSP. Bianchini et al [10] categorized the QoS information into three types, namely numeric, Boolean and enumeration and the users' QoS requirement of each type can be Good, Poor and Fair etc. The service discovery process is performed by matching those categories. Giallonardo and Zimeo [17] provided an ontology-based matching approach to avoid one-by-one QoS parameter matching failures which are caused by misunderstanding the terminology of QoS attributes.

3 QoS Information Description

This section presents a novel model to represent both service providers' QoS advertisements and service consumers' QoS requirements. This model is generic and can easily be extensible to provide comprehensive and explicit description of QoS information.

3.1 QoS Information Advertisement

A service provider can conventionally publish its service information such as businessEntity, businessService, and bindingTemplate. In order to publish its QoS capabilities, a service provider needs to include an entity called QoSInformation in its advertisement. As shown in Figure 1, the QoSinformation can contain information of one or more QoS attributes, each of which is described using a five-tuple, <*attributeName*, *attributeType*, *attributeValue*, *attributeUnit*, *constraints*>.

The element *attributeName* represents the name of the QoS attribute, such as price, response time, availability, and so on. The element *attributeType* represents the type of the QoS attribute. In this research, three types of QoS attributes are introduced, including *Numeric QoS attribute*, *Boolean QoS attribute*, and *Enumeration QoS attribute*. In the matching process, the different algorithm will be invoked according to the type of attribute. The element *attributeUnit* defines the measurement units of the QoS attribute. For example, the *attributeUnit* of the attribute price can be DOLLAR and the *attributeUnit* of the attribute response time can be DAY or HOUR. The element *attributeValue*, represented in different forms according to the type of the QoS attribute, defines the advertised value of the QoS attribute. For Numeric QoS attribute, two numeric values describe the value range of this attribute as an interval. For Boolean QoS attribute, the *attributeValue* can be either TRUE or FALSE. For Enumeration QoS attribute, the *attributeValue* is a discrete value set. The element *constraint* allows service providers to publish the constraints of their services, such as the

server maintenance time, upgrade information, price adjustment and so on. The following sample advertisement <execution time, numeric, [0, 3], hour, (between 9am and 5pm, Monday to Friday)> describes a service provider's QoS promises on attribute execution time. From this advertisement, it is clear that the execution time of this service is no more than 3 hours between 9am and 5pm, Monday to Friday.

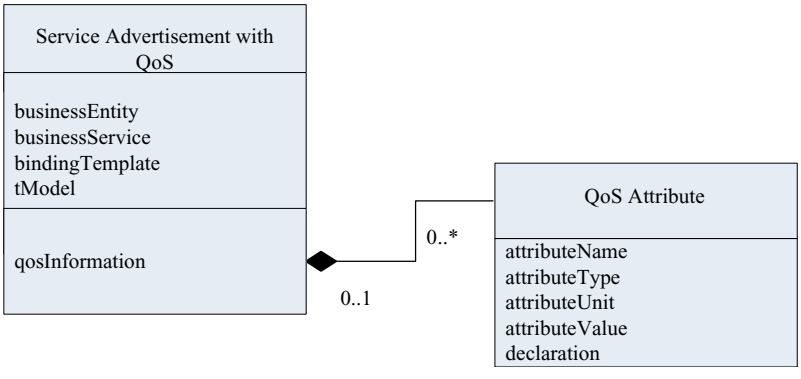


Fig. 1. Structure of QoS Advertisement

3.2 Storage of QoS information

Service providers' QoS information needs to be embedded in the current UDDI framework for it to be used in discovery. To achieve this, a new tModel called QoS Information tModel is created, as depicted in Figure 2. The element overviewURL of this tModel is referred to an external file which stores all the QoS information about this service. Such an approach requires minimum modification to the current UDDI data structure. The external file can be hosted by a third party or even by the service provider.

```
- <tModel tModelKey="uuid:xxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx">
    <name>qualityInformation</name>
    <description>Quality of Service Information</description>
  - <overviewDoc>
      <description xml:lang="en" />
      <overviewURL>externalQoSURL</overviewURL>
    </overviewDoc>
    <categoryBag>....</categoryBag>
  </tModel>
```

Fig. 2. QoS Information tModel

3.3 QoS Requirement

In order for a service consumer to discover "right" services, the service discovery request should contain information to represent the QoS requirements of the service consumer, in addition to functional requirements. As shown in Figure 3, a service consumer can include requirements on one or more QoS attributes in the service discovery request. The requirement on a QoS attribute can either be compulsory or optional, which will be treated differently in matching and ranking.

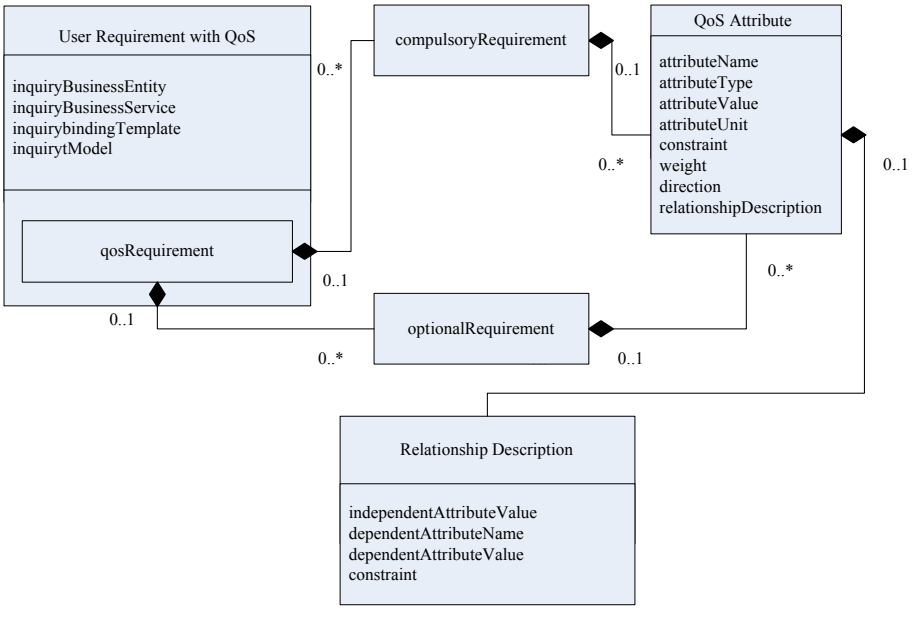


Fig. 3. Structure of QoS requirement

A compulsory requirement attribute is described by the following eight-tuple:

<*attributeName*, *attributeType, attributeValue*, *attributeUnit*, *constraint*, *weight*, *direction, relationship*>

The first four elements are of the same meanings as those discussed in Section 3.1. The element *constraint* which can be either *belongTo or notBelongTo* is introduced to illustrate the restriction of the *attributeValue*. For instance, assuming that a user does not need a service between 1pm to 3pm, this requirement can be described by this model, using an interval [1, 3] and a *constraint* value *notBelongTo*. The element *weight* indicates the importance of this attribute to the service consumer. The value of *weight* should be a number between 0 and 1 and it should be assigned by the consumer. This attribute is introduced to rank the services in the ranking process. The element *direction* only used to describe numeric attributes to show the expected tendency of the attribute. The *positive* direction means that user expects an increasing trend of this numeric attribute, whereas *negative* shows that user expects a decreasing trend of this attribute. Both *weight* and *direction* are used in the ranking algorithm discussed in Section 4 to compare discovered services. Finally, the element *relationship* is used to describe the complex requirements on the relationship of two QoS attributes.

The relationships between attributes are used to trade-off or negotiate with the service discovery system. This paper only provides a way to describe the relationship between two compulsory attributes. In a statement of a relationship, the related two attributes are treated as piecewise function. The value of the dependent attribute is contingent on the independent attribute. Furthermore, the omission of *independentAttributeName* and *independentAttributeType* is because these two attributes map with

```
<?xml version="1.0" encoding="UTF-8"?>
<qosRequirement>
  <compulsoryRequirement>
    <qosAttribute name = "availability">
      <attributeName>availability</attributeName>
      <attributeType>numeric</attributeType>
      <attributeValue>0,99.99 </attributeValue>
      <attributeUnit>percent</attributeUnit>
      <constraint>belongTo</constraint>
      <weight>0.2 </weight>
      <direction>positive</direction>
      <relationship>
        <independentAttributeValue>99.99,99.9999 </independentAttributeValue>
        <dependentAttributeName>encryption</dependentAttributeName>
        <dependentAttributeValue>false</dependentAttributeValue>
        <constraint></constraint>
      </relationship>
    </qosAttribute >
    <qosAttribute name = "encryption">
      <attributeName>encryption</attributeName>
      <attributeType>boolean</attributeType>
      <attributeValue>true</attributeValue>
      <constraint>belongTo</constraint>
      <weight>null</weight>
      <direction>null</direction>
      <relationship>null</relationship>
    </qosAttribute>
  </compulsoryRequirement>
  <optionalRequirement>
    <attributeName>resolution</attributeName>
    <attributeType>enumeration</attributeType>
    <attributeValue>600,800,1024,768,1680,1050 </attributeValue>
    <attributeUnit>pixel</attributeUnit>
    <constraint>belongTo</constraint>
    <weight>0.8</weight>
    <direction>null</direction>
  </optionalRequirement>
</qosRequirement>
```

Fig. 4. A Sample of QoS Requirement

the content of the *attributeName* and *attributeType* which are located in the description of the attribute. Similarly, it is not necessary to reclaim the *dependentAttributeType* because it is mapped to the attribute according to the *dependentAttributeName*.

3.4 A Sample of QoS Requirement

In this section, the data structure of the proposed QoS requirement is realized by a XML document. In the sample shown in Figure 4, the user has two compulsory requirements about availability and resolution and an optional requirement about encryption. Moreover, the requirement includes a relationship between availability

and resolution, namely while the availability more than 99.99 percent and less than 99.9999percent then the resolution can be 600×800.

4 Matching and Ranking Algorithms

The matching algorithm is to locate the services with matching QoS information, whereas ranking algorithm is in charge of present them to a consumer in an order that may best reflect the consumer' interests.

4.1 Matching Algorithm

Given a service request R which contains compulsory requirements on n QoS attributes A_1, A_2,... A_n, the objective of the matching algorithm is to select, among those services that provide the required functions of R, services that satisfy these QoS requirements. The strategy is to examine these requirements one by one to obtain service sets S_1, S_2,... S_n which represent the set of services that satisfy n compulsory QoS requirements, respectively. For a compulsory requirement on a numeric type attribute, the upper bound and lower bound of the interval will be compared between the requirement and the advertisement respectively. If the upper bound of the requested interval is larger than the upper bound of published interval and the lower bound of requested interval is smaller than the lower bound of published interval then this service should be selected, otherwise this service should be discarded. For a compulsory requirement on a Boolean type attribute, if the value of a requested Boolean attribute matches with the published Boolean attribute then this service should be selected, otherwise, the service should be discarded. For a compulsory requirement on an enumeration type attribute, if the published enumeration attribute set E_p is a subset of the request attribute set E_R, then the service should be selected, otherwise, the service should be discarded. Finally, the result of the matching service set *S* is calculated by the following equation.

$$S = \bigcap_{i=1}^{n} S_i \tag{1}$$

The Detailed Matching Algorithm Is Shown in Figure 5.

Using symbol S_{R_i} to represent the service set that satisfies the relationship statements of the attribute A_1; whereas S_C indicates the service set that satisfies the *constraint* statement of this attribute. The service set S_1 should be the union of S_{R_1} and S_{C_1}. If there are multiple relationships, the formula can be summarized as follows:

$$S_i = (\bigcup_{i=1}^{n} S_{R_i}) \bigcup S_C \tag{2}$$

Input: serviceSet ($S_0, S_1, ..., S_n$)

Input: requirement

1. **for** (i = 0, i < n, i++)
2. **if** ($U_R \geq U_P \parallel L_R \leq L_P$) // The U_R indicates the upper bound of //requirement interval, whereas U_P indicates the upper bound of published //service interval.
3. **return** $S_{Numeric} = S_n$ // Retrieve all the services which satisfy the //numeric attribute.
4. **else** discard S_n
5. **end if**
6. **end for**
7. **for** (i = 0, i < n, i++)
8. **if** ($B_R \equiv B_P$) // The B_R indicates the Boolean attribute of requirement, //whereas B_P indicates Boolean attribute of published service.
9. **return** $S_{Boolean} = S_n$ // Retrieve all the services which satisfy the //Boolean attribute.
10. **else** discard S_n
11. **end if**
12. **end for**
13. **for** (i = 0, i < n, i++)
14. **if** ($E_R \supset E_P$) // The E_R indicates the enumeration attribute of //requirement, whereas E_P indicates enumeration attribute of published service.
15. **return** $S_{Enumeration} = S_n$ // Retrieve all the services which satisfy the //enumeration attribute.
16. **else** discard S_n
17. **end if**
18. **end for**
19. $S = S_{Numeric} \cap S_{Boolean} \cap S_{Enumeration}$ // Final result of matching algorithm.

End algorithm

Fig. 5. Pseudocode of Matching Algorithm

Thus, combining formulas (1) and (2), the final matching formula becomes:

$$S = \bigcap_{i=1}^{n} ((\bigcup_{i=1}^{n} S_{R_i}) \bigcup S_C) \tag{3}$$

4.2 Ranking Algorithm

If the matching algorithm returns more than one matched service, it is a desirable feature to rank these services in an order that best represents the consumer's preference. The ranking measures the "goodness" of the matched services and makes recommendation to the consumers according to their needs.

The ranking algorithm takes both the compulsory and optional QoS requirements into consideration. In order to compute the preference of users, QoS metrics are introduced in this approach. Firstly, all of the published QoS attributes of service p are modeled as a vector $A_p = \{A_{p,1}, A_{p,2}, ..., A_{p,n}\}$ (n is the number of attributes) Secondly, the QoS requirements in the service request can be modeled as a n-dimension binary vector as well, namely, $A_r = \left((A_1, w_1), (A_2, w_2), ..., (A_n, w_n)\right)$, where $\sum_{i=1}^{n} w_i = 1, w_i \in [0,1]$. w_i is the weight of the attribute i which is normally given by the service consumer. Thirdly, a distance $dis(A_p, A_r)$ between the requested

vector A_r and the published vector A_p could be calculated. It is believed that a service that is more close to the requirements would satisfy the consumer better.

Based on the above, the situation that there is *m* services that each contains n QoS attributes can be modeled as an *m*×*n* matrix Q :

$$Q = \begin{matrix} A_{1,1} & A_{1,2} & \dots & A_{1,n} \\ A_{2,1} & A_{2,2} & \dots & A_{2,n} \\ \dots & \dots & \dots & \dots \\ A_{m,1} & A_{m,2} & \dots & A_{m,n} \end{matrix}$$

Then, the normalization process should be applied. This process is conducted according the value of direction as well. Each element in the Q matrix should be normalized into the interval [0, 1]. If the direction is positive, then the formula (4) will be used to normalize. In contrast, if the direction is negative, the formula (5) will be used.

$$Nor(A_{i,k}) = \frac{\max A_{i,k} - A_{i,k}}{\max A_{i,k} - \min A_{i,k}} \text{ (i = 1, 2,… m); (k = 1, 2,… n)} \quad (4)$$

$$Nor(A_{i,k}) = \frac{A_{i,k} - \min A_{i,k}}{\max A_{i,k} - \min A_{i,k}} \text{ (i = 1, 2,… m); (k = 1, 2, … n)} \quad (5)$$

After that, matrix Q should be merged with the requirement vector A_r to get a *(m+1)* ×*n* matrix Z .

$$Z = \begin{vmatrix} A_{r_1} & A_{r_2} & \dots & A_{r_n} \\ A_{1,1} & A_{1,2} & \dots & A_{1,n} \\ A_{2,1} & A_{2,2} & \dots & A_{2,n} \\ \dots & \dots & \dots & \dots \\ A_{m,1} & A_{m,2} & \dots & A_{m,n} \end{vmatrix}$$

The requirement vector A_r should be optimized in order to indicate a QoS destination in the QoS space, namely all of the attributes of requirement adopt the best value. Rather, *Direction* represents the user preference tendency, the optimized numeric value should be the biggest or the smallest accordingly. For the Boolean and enumeration type attributes, the optimized value should be 1 which is used to indicate the meaning of "true".

Finally, using the formula (6) to calculate the distance between A_p and A_r .

$$dis(A_p, A_r) = \sqrt{\sum_{i=1}^{m} W(A_{i,j} - A_{r_j})^2} \quad (6)$$

Comparing the distance $dis(A_p, A_r)$, the ranking list can be generated. If the requested optional QoS attribute is absent within the register service then set the distance as 1.

5 Conclusion and Future Work

The lack of consideration of non-functional requirements in service discovery may lead to the problem that the retrieved services cannot guarantee the quality of the services. This paper made an important step towards fully functional QoS-based service discovery by proposing a novel approach to extending UDDI with consideration of QoS, including QoS information description, QoS information storage and QoS information discovery. More specifically, this paper has established a XML data structure to represent QoS information of both service providers and service consumers. A QoS information tModel has been introduced to reference these QoS information documents. In addition, a QoS matching algorithm and a QoS ranking algorithm have been presented to make use of the QoS information in service discovery.

In the future, further research work will be carried out in QoS-based service discovery. The priority will be given to an implementation of the proposed approach for the purpose of proof-of-concept. Based on this implementation, accuracy and efficiency of the matching and ranking algorithms will be evaluated, followed by improvement. In addition, validation of service consumers' discovery request to ensure the consistency between QoS requirements will be investigated.

Acknowledgement

This research is partially supported by University of Wollongong Small Research Grant.

References

1. Shuping, R.: A Model for Web Services Discovery with QoS. SIGecom Exch. 4(1), 1–10 (2003)
2. Andrea, D.A.: A Model-driven WSDL Extension for Describing the QoS of Web Services. In: Proc. of 2006 IEEE International Conference on Web Services (ICWS 2006), Chicago, USA, pp. 789–796 (September 2006)
3. Clement, L., Hately, A., von Riegen, C., Rogers, T.: UDDI Version 3.0.2, OASIS (October 19, 2004), `http://uddi.org/pubs/uddi_v3.htm`
4. Kritikos, K., Plexousakis, D.: A Semantic QoS-based Web Service Discovery Algorithm for Over-Constrained Demands. In: Proc. of the 3rd International Conference on Next Generation Web Services Practices (NWeSP 2007), Korea, 29-31 October 2007, pp. 49–54 (2007)
5. Duwaldt and Trees. Web Services A Technical Introduction, DEITEL™ Web Services Publishing. GUNTHER, N.J, The Practical Performance Analyst. McGraw-Hill.ISO, New York (1998), `http://www.iso.ch/iso/en/ISOOnline.frontpage`

6. Chen, Z., Liang-Tien, C., et al.: DAML-QoS ontology for Web services. In: Proceedings. IEEE International Conference on Web Services, San Diego, California, USA, 6-9 July 2004, pp. 472–479 (2004)
7. Maximilien, E.M., Singh, M.P.: A framework and ontology for dynamic Web services selection. Internet Computing 8(5), 84–93 (2004)
8. Dobson, G., Lock, R., et al.: QoSOnt: a QoS ontology for service-centric systems. In: 31st EUROMICRO Conference on Software Engineering and Advanced Applications, Porto, Portugal, 30 August -3 September 2005, pp. 80–87 (2005)
9. Gwyduk, Y., Taewoong, Y., et al.: A QoS model and testing mechanism for quality-driven Web services selection. In: The Fourth IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, 2006 and the 2006 Second International Workshop on Collaborative Computing, Integration, and Assurance. SEUS 2006/WCCIA 2006, 27-28 April 2006, p. 6 (2006)
10. Bianchini, D., De Antonellis, V., et al.: QoS in ontology-based service classification and discovery. In: Proceedings of the 15th International Workshop on Database and Expert Systems Applications, Washington, DC, USA (2004)
11. Blum, A., Carter, F.: Representing Web Services Management Information, http://www.oasisopen.org/committees/download.php/5144/
12. Yang, L., Huacan, H.: Grid Service Selection Using QoS Model. In: Third International Conference on Semantics, Knowledge and Grid, Xi'an, China, 29-31 October 2007, pp. 576–577 (2007)
13. Somasundaram, T.S., Balachandar, R.A., et al.: Semantic Description and Discovery of Grid Services using WSDL-S and QoS based Matchmaking Algorithm. In: ADCOM 2006. International Conference on Advanced Computing and Communications, 20-23 December 2006, pp. 113–116 (2006)
14. Kyriakos, K., Dimitris, P.: Semantic QoS Metric Matching. Web Services. In: 4th European Conference on ECOWS 2006 (2006)
15. Wenli, D.: QoS Driven Service Discovery Method Based on Extended UDDI. In: Third International Conference on Natural Computation, ICNC 2007, 24-27 August 2007, pp. 317–324 (2007)
16. D'Mello, D.A., Ananthanarayana, V.S., et al.: A QoS Broker Based Architecture for Dynamic Web Service Selection. In: Second Asia International Conference on Modeling & Simulation, AICMS 2008 (2008)
17. Yu, T., Lin, K.J.: The design of QoS broker algorithms for QoS-capable Web services. In: 2004 IEEE International Conference on e-Technology, e-Commerce and e-Service, pp. 17–24. IEEE, Los Alamitos (2004)
18. Giallonardo, E., Zimeo, E.: More Semantics in QoS Matching. In: IEEE International Conference on Service-Oriented Computing and Applications, SOCA 2007, Newport Beach, California, USA, June 19-20 (2007)
19. Mani, A., Nagarajan, A.: Understanding quality of service for Web services (2002), http://www-106.ibm.com/developerworks/library/ws-quality.html
20. Yu-jie, M., Jian, C., Shen-sheng, Z., Jian-hong, Z.: Interactive Web Service Choice-Making Based on Extended QoS Model. Journal of Zhejiang University - Science A 7(4), 483–492 (2006)

A Redundancy Protocol for Service-Oriented Architectures

Nicholas R. May

RMIT University, Melbourne, Australia
Nicholas.May@rmit.edu.au

Abstract. Achieving high-availability in service-oriented systems is a challenge due to the distributed nature of the architecture. Redundancy, using replicated services, is a common software strategy for improving the availability of services. However, traditional replication strategies are not appropriate for service-oriented systems, where diverse services may be grouped together to provide redundancy. In this paper we describe the requirements for a redundancy protocol and propose a set of processes to manage redundant service providers.

1 Introduction

Service-Oriented Architecture (SOA) is a style of software architecture that promotes software reuse and inter-operability. This it achieves by distributing its functionality amongst services, which are loosely coupled software components. However, the distributed nature of SOA presents a serious challenge to a system's quality of service when services are spread across organizational boundaries.

The availability of a service is a quality that is difficult to manage when the service depends on inter-organizational services. It is usually improved by using redundancy, in the form of additional components that provide backup services in the event of a failure. Traditional redundancy strategies are principally concerned with the synchronization of state between identical components, but services in an SOA system are independent and autonomous and so do not need synchronizing. However, some service invocations incur a cost or result in a change in the shared state. Therefore, an SOA redundancy strategy is required to ensure that only one redundant service is executed per invocation.

We address this problem by identifying the requirements that a process must meet if it is to manage redundant services. These requirements are met by adapting the three-phase commit (3PC) fault tolerance protocol to SOA.

In the following sections we will cover some background, the protocol, and draw some conclusions about the protocol and its limitations.

2 Background

In order to define a protocol for redundancy in SOA, we must first provide some background in the relevant subject areas. In this section we discuss the areas of service-oriented architecture, fault tolerance, redundancy, and related work.

G. Feuerlicht and W. Lamersdorf (Eds.): ICSOC 2008, LNCS 5472, pp. 211–220, 2009.

2.1 Service-Oriented Architecture

Software architecture is a discipline of software engineering. It can also be viewed as an abstraction of a software system, in terms of its functional components, the properties of the components, and the relationships between them [1].

The software components provide the systems functionality and can range from a primitive computational unit to a whole composite systems. The relationships between components are called connectors. They represent the means of communication between components, and can represent a simple association or a more complex interaction. The properties of a component are its externally visible, non-functional qualities. They influence the quality by which the functionality is provided by the component, but do not affect the functionality provided. Availability is an example of a non-functional property.

Another feature of architecture is the use of styles. These are patterns of software composition that have well known quality consequences. Styles are defined by the components types, their relations and the rules by which they may be combined [18].

Service-Oriented Architecture (SOA) is a style of software architecture designed to utilize distributed components that may be located across organizational boundaries. Its goals are to promote the reuse, evolution, scalability and interoperability of software components [12]. SOA components are called services, which interact through the publish and subscribe connection pattern.

The principles of SOA provide guidelines for implementing systems so that the aims of SOA can be achieved [2,5,12,15,20]. The important principles for this study are as follows;

- **Discoverability:** Services should be visible, such that they can be found and accessed via a discovery mechanism. This is accomplished by publishing descriptions, to some form of repository, in a widely accessible and understandable format.
- **Composability:** Services can be composed into composite services, either by static definitions or by the dynamic discovery of services at run time.
- **Statelessness:** The purpose of invoking a service is to realize an effect. This may be a response message or a change in the shared state of the participating services. However, a service provider is stateless in that it need not retain information about the state of a service consumer between invocations.

Other features of a service include; defined service contract, loose coupling, autonomy, abstraction, and reusability.

SOA is an abstract architecture in that it only describes the principles to which service-oriented systems should adhere. A common set of standards by which these systems can be implemented, collective known as Web Services, are published by the Organization for the Advancement of Structured Information Standards (OASIS) and the World Wide Web Consortium (W3C). The core standards provide for the description, publication, discovery, and consumption of services using the publish-subscribe communication paradigm.

Additional specifications, commonly known as the WS-* extensions, define standards for managing the quality of service consumption. Some of the Web Service extensions include; WS-ReliableMessaging (OASIS), WS-Coordination (OASIS), WS-AtomicTransaction (OASIS), and WS-Policy (W3C). These provide frameworks for managing messages, coordinating actions, implementing fault tolerance features, and defining policies for quality of service constraints. Two additional languages have been defined to allow the specification of orchestrations, BPEL4WS (OASIS), and choreographies, CDL4WS (W3C), of interacting services.

2.2 Fault Tolerance

A fault tolerant system aims to avoid system failure even if faults are present. In terms of SOA, a failure occurs if a service is unable to respond to a request as defined by its published definition. In an inter-organization SOA the most important phase of fault tolerance is error detection. Once an error is detected, a service consumer must take the appropriate action to find another provider to fulfill its request. This form of error recovery is the only guaranteed option available, where failed service providers may be in external domains. If no alternatives are available for a critical service provider then the fault cannot be recovered and the consumer must propagate the failure. The service will remain in a failure state until it can discover a working service provider to satisfy its critical functions.

Many techniques have been proposed to improve the fault tolerance of distributed systems. These can be characterized as either optimistic or conservative approaches. Optimistic techniques make assumptions from the properties of the system in order to improve the performance of fault tolerance. However, when the assumptions fail the technique requires additional work to undo operations. A discussion of optimistic approaches is provided by Jiménez-Peris and Patiño-Martínez [8].

Conservative techniques, such as atomic commitment, involve a greater number of messages under normal operation than an optimistic technique, and hence a worse performance. Two examples of atomic commitment are the two-phase (2PC) and three-phase (3PC) commit protocols. Both protocols assume that the communications network is reliable and detection of service failures is identified by timeout actions initiated by the non arrival of expected messages. However, the 3PC contains additional operations and states that ensure that it is a non-blocking protocol.

A finite state automata (FSA) of a 3PC protocol, adapted from Jalote [7, p239], is shown in Fig. 1. This FSA shows the input and output messages associated with state transitions, which synchronize a COORDINATOR with any number of PARTICIPANT components. It can seen that this is a non-blocking protocol because there is no commit state (c) adjacent to an abort state (a) or non-committable state.

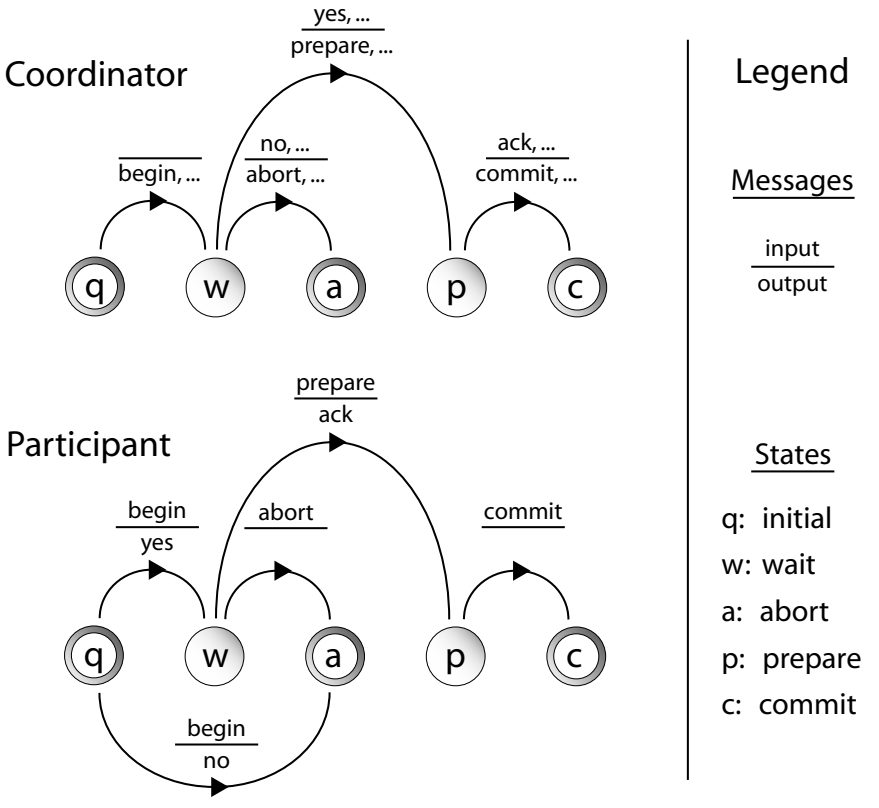


Fig. 1. FSA of a Three-Phase Commit protocol

2.3 Redundancy

An availability of less than 100% can result in a failure of a system if it means that a critical call cannot be serviced. If the required functionality is available in another component then a system failure may be avoided. A common strategy for improving the availability of a software component involves adding redundancy into the functionality on which the component depends [17]. Redundancy can be defined as the introduction of components that are not needed for the correct operation of the system if no failures occur [7]. This can be achieved by replicating copies of critical components.

In SOA, replication strategies must be assessed in terms of the service definition and the dynamic nature of binding. Service discovery allows a consumer to build a list of providers that are not identical but can still satisfy the required contract. These can be considered replicas for the sake of redundancy in SOA. However, these replicas may exist across organizational boundaries, so any strategy that relies on communication between replicas must be excluded, which means that passive replication [7] is not feasible in an SOA system. Furthermore,

we can assume that any request to a group of replicas is atomic, stateless and satisfied if "at-least-one" response is returned. Finally, the distributed nature of an inter-organization SOA means that functional error detection is not guaranteed. Therefore, the best detection strategy is a time check failure, such as failing on timeout of a response.

2.4 Related Work

Studies of service redundancy have been focused on traditional solutions using service replication. They can generally be divided into replication strategies, replication architectures, and implementation frameworks.

Several studies that focus on the various replication strategies, such as 'active' and 'passive' techniques, are presented by Maamar et al. [11], Guerraoui and Shipper [6], and Chan et al. [3]. These papers discuss replication communities in Web Services, survey replication techniques, and evaluate temporal and spacial redundancy techniques.

Architectures are described by Osrael et al. [14], who propose a generalized architecture for a service replication middleware, and Juszczyk et al. [9], who describe a modular replication architecture.

Among the frameworks are those described by Salas et al. [16], Engelmann et al. [4], and Laranjeiro and Vieira [10]. They propose an active replication framework for Web Services, a virtual communication layer for transparent service replication, and a mechanism for specifying fault tolerant compositions of web services using diverse redundant services, respectively.

The focus of these studies is on describing and implementing various strategies for invoking and maintaining replicated services. However, most do not treat services as autonomous components, and so their applicability is limited for an SOA system.

3 Protocol

A protocol is a set of rules governing the exchange of data between devices [19]. In this instance, it can be described by the processes, states and allowable actions, that satisfy the protocol's requirements. In order to deduce a protocol we must first define the requirements and assumptions for redundancy in an SOA system.

We can make the following assumptions of services included in a redundancy group. Service are stateless outside of the protocol. Each conversation with an available provider is assumed to be reliable because there are existing protocols, such as WS-ReliableMessaging and WS-AtomicTransaction, to manage a conversation once it is established. Finally, operations are not guaranteed to be idempotent, in that they may have an associated cost or state change for each invocation. However, any operation that is idempotent, such as a simple query, may be invoked many times without consequence and so will not require a redundancy protocol.

```
CANDIDATE  = (begin    -> WAIT),
WAIT       = (no       -> END     | yes      -> READY),
READY      = (abort    -> END     | prepare  -> PREPARED),
PREPARED   = (execute  -> END).
```

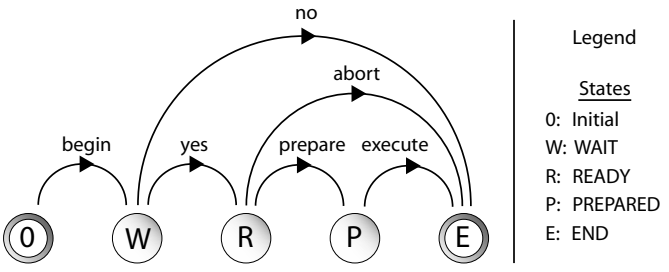


Fig. 2. FSP and state machine of the CANDIDATE process

The requirements of the protocol are as follows;

1. Provider services are autonomous (SOA).
2. Provider services in a redundancy group may have different contracts, but each must satisfy a common sub-contract (SOA).
3. Consumers must be able to conduct multiple, simultaneous conversations with provider services (Fault Tolerance).
4. Any invocation of an operation that has a cost must result in the execution of only one redundant service (SOA).
5. Providers are selected by voting or time ordering (Fault Tolerance).

These requirements lead to the following attributes of the protocol. Firstly, each redundant service must be modeled as an independent concurrent process (Req. 1). Secondly, the protocol must be independent of a particular service contract (Req. 2). Thirdly, the protocol must be non-blocking (Req. 3). Fourthly, the protocol must support 'at most once' execution (Req. 4). Finally, the protocol must include a controlling action that can select which redundant service to invoke (Req. 5). A solution to these requirements is to adapt the 3PC protocol. This must be modified to ensure that only one process is executed, rather than all committed. To reflect this change in emphasis the name 'Participant' is replaced by the name 'Candidate' to represent the redundant processes.

The constituent processes of the protocol are modeled as Finite State Processes (FSPs) [13]. FSP is a language especially suited to modeling synchronized, concurrent processes. Processes are defined in a textual language that represents the states and the actions that trigger state transitions. Concurrency is modeled in FSP with interleaved actions. However, actions that must be performed simultaneously can be defined with shared action pairs. FSP does not show message exchange explicitly, but messages are the normal mechanism used to implement shared actions. The Labeled Transition System Analyzer (LTSA) [13] is a tool that can be used to generate state machines from FSP definitions and to check

```
COORDINATOR  = (begin    -> WAIT),
WAIT         = (timeout  -> END     | found   -> READY),
READY        = (timeout  -> END     | select  -> PREPARED),
PREPARED     = (invoke   -> END).
```

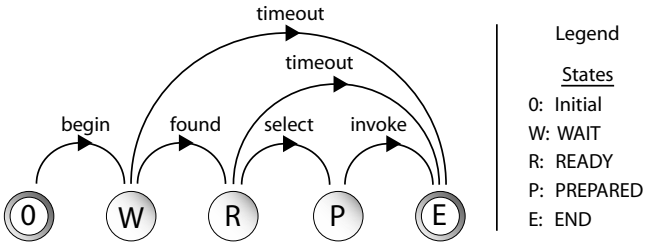


Fig. 3. FSP and state machine of the COORDINATOR process

their properties. The protocol definition using FSP consists of a three processes; CANDIDATE, COORDINATOR and REDUNDANCY.

The CANDIDATE process is initialized with a *begin* action, after which it will respond with a *yes* or *no* action to indicate whether it is able to perform its service. If it is able, it will wait until it receives a *prepare* action. If none is received before a specified timeout period then it will *abort* the process. A CANDIDATE that is PREPARED will then be *executed*. This process satisfies the requirements for a non-blocking protocol because the *execute* and *abort* actions are not available from the same state. The FSP and State Machine for the CANDIDATE process are shown in Fig. 2.

The COORDINATOR process is also initialized with a *begin* action. It will then wait until it receives a message to indicate that a CANDIDATE has been *found*. If none are found before a specified period then the COORDINATOR will *timeout* and the process will end. If a CANDIDATE is found then the COORDINATOR will *select* an appropriate CANDIDATE using a specified election method, for instance the first response. Finally, the COORDINATOR will *invoke* the CANDIDATE in the PREPARED state and accept the response. Similarly to the CANDIDATE process, this process also satisfies the requirements for a non-blocking protocol. The FSP and State Machine for the COORDINATOR process are shown in Fig. 3.

The REDUNDANCY process consists of a COORDINATOR and two CANDIDATE processes, labelled *a* and *b*. In addition, the FSP includes the following shared actions pairs; *(begin,begin)*, *(yes,found)*, *(select,prepare)*, and *(invoke,execute)*. This process ensures that the COORDINATOR will select and invoke a single CANDIDATE, and the other CANDIDATE processes will abort if available but not selected. The FSP and state machine for a REDUNDANCY process with two CANDIDATE processes is shown in Fig. 4. The state machine is shown only to give an impression of its scale. This process satisfies the requirements for a redundancy protocol in an SOA system because;

- Each service provider is modeled by a separate CANDIDATE process.
- The protocol is independent of any particular service contract

```
||REDUNDANCY = (COORDINATOR || {a,b}:CANDIDATE)
                /{ begin/{a,b}.begin,
                   {a,b}.yes/found,
                   {a,b}.prepare/select,
                   {a,b}.execute/invoke }.
```

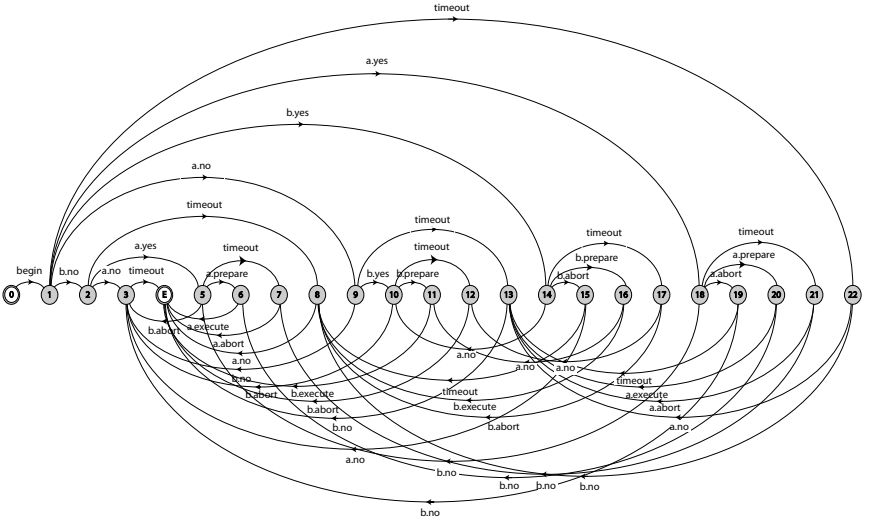


Fig. 4. FSP and state machine of a REDUNDANCY process with two CANDIDATES

- The processes are non-blocking.
- "At most one" CANDIDATE is executed.
- The 'select' action provides a mechanism for the COORDINATOR to determine which CANDIDATE to invoke.

The state machine covers all available paths of the interleaved actions, synchronized by the shared actions. The safety and liveness properties of the state machine can be validated using the LTSA. Safety properties include the absence of deadlocks and mutual exclusion of participant execution. The LTSA reports no deadlocks for the state machine. In addition, the LTSA animator allows individual paths to be traced through the state machine. This shows that each path that includes an *execute* action includes only one. Liveness properties include path progress and successful termination. The LTSA shows that all states, except the END state, have at least one out action, and that all paths eventually terminate at the END state.

4 Conclusions

In this paper we discuss the background to redundancy in SOA systems and identified the requirements for a protocol to manage redundant services which

are not idempotent. These requirements have been satisfied by adapting the three-phase commit protocol to ensure that 'at most one' redundant service is executed per invocation. The protocol consists of two non-blocking processes (CANDIDATE and COORDINATOR) and a synchronizing process (REDUNDANCY), all of which have been modeled as finite state processes.

The protocol is a conservative, passive, fault detection mechanism, in that it cannot predict where a fault will occur or actively exclude unavailable services before the invocation process begins. This will reduce the performance and increase the number of messages at invocation compared to an active or optimistic protocol. However, a conservative protocol can be adapted to combine the fault detection with the negotiation of quality attributes for the provided service. Therefore, this protocol would be more efficient in a dynamic, quality negotiation scenario.

Only a simple redundancy process has been modeled, with two redundant candidates and selection by first response. This protocol would benefit by modeling more complex redundancy groups and investigation of how to integrate different selection strategies. In addition, the modeling of fault recovery has not been considered. For instance, how would the protocol recover from a failure whilst in the *prepared* state?

The processes of this protocol have not yet been implemented. Further investigation will be required to determine the most appropriate method to communicate and implement different redundancy strategies. The various Web Service specifications may provide a basis for the protocol and a means to communicate implementation parameters, such as whether a service invocation incurs a cost.

References

1. Bass, L., Clements, P., Kazman, R.: Software Architecture in Practice, 2nd edn. Addison-Wesley, Reading (2003)
2. Bastani, F., Ma, H., Gao, T., Tsai, W.-T., Yen, I.-L.: Toward qos analysis of adaptive service-oriented architecture. In: IEEE International Workshop on Service-Oriented System Engineering (SOSE 2005), pp. 219–226 (2005)
3. Chan, P.P.W., Lyu, M.R., Malek, M.: Making services fault tolerant. In: Penkler, D., Reitenspiess, M., Tam, F. (eds.) ISAS 2006. LNCS, vol. 4328, pp. 43–61. Springer, Heidelberg (2006)
4. Engelmann, C., Scott, S.L., Leangsuksun, C., He, X.: Transparent Symmetric Active/Active Replication for Service-Level High Availability. In: 7th IEEE International Symposium on Cluster Computing and the Grid, Rio de Janeiro, Brazil, May 2007, pp. 14–17 (2007)
5. Erl, T.: Service-Oriented Architecture: Concepts, Technology, and Design. The Prentice Hall Service-Oriented Computing Series from Thomal Erl. Prentice Hall, Upper Saddle River (2005)
6. Guerraoui, R., Schiper, A.: Software-based replication for fault tolerance. Computer 30(4), 68–74 (1997)
7. Jalote, P.: Fault Tolerance in Distributed Systems. PTR Prentice Hall, Englewood Cliffs (1994)

8. Jiménez-Peris, R., Patiño-Martínez, M.: Towards Robust Optimistic Approaches. In: Schiper, A., Shvartsman, M.M.A.A., Weatherspoon, H., Zhao, B.Y. (eds.) Future Directions in Distributed Computing. LNCS, vol. 2584, pp. 45–50. Springer, Heidelberg (2003)
9. Juszczyk, L., Lazowski, J., Dustdar, S.: Web service discovery, replication, and synchronization in ad-hoc networks. In: 1st International Conference on Availability, Reliability and Security (ARES 2006), pp. 847–854. IEEE Computer Society, Washington (2006)
10. Laranjeiro, N., Vieira, M.: Towards fault tolerance in web services compositions. In: 2007 workshop on Engineering fault tolerant systems (EFTS 2007), p. 2. ACM, New York (2007)
11. Maamar, Z., Sheng, Q.Z., Benslimane, D.: Sustaining web services high-availability using communities. In: Third International Conference on Availability, Reliability and Security (ARES 2008), Barcelona, Spain, pp. 834–841 (March 2008)
12. MacKenzie, C.M., et al.: Reference Model for Service Oriented Architecture 1.0. Organization for the Advancement of Structured Information Standards (October 2006) (October 5, 2007), http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.html
13. Magee, J., Kramer, J.: Concurrency: State Models and Java Programming, 2nd edn. John Wiley and Sons, Chicester (2006)
14. Osrael, J., Froihofer, L., Goeschka, K.M.: What service replication middleware can learn from object replication middleware. In: 1st Workshop on Middleware for Service Oriented Computing (MW4SOC 2006), pp. 18–23. ACM, New York (2006)
15. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F., Krämer, B.J.: Service-oriented computing: A research roadmap. In: Cubera, F., Krämer, B.J., Papazoglou, M.P. (eds.) Service Oriented Computing (SOC), Schloss Dagstuhl, Germany. Dagstuhl Seminar Proceedings, vol. 05462, Internationales Begegnungs und Forschungszentrum fuer Informatik (IBFI) (2006)
16. Salas, J., Pérez-Sorrosal, F., Patiño Martínez, M., Jiménez-Peris, R.: WS-Replication: a framework for highly available web services. In: 15th International Conference on World Wide Web (WWW 2006), pp. 357–366. ACM, New York (2006)
17. Schmidt, K.: High Availability and Disaster Recovery: Concepts, Design, Implementation. Springer, Berlin (2006)
18. Shaw, M., Garlan, D.: Software Architecture: Perspectives on an Emerging Discipline. Prentice-Hall, Upper Saddle River (1995)
19. Thompson, D. (ed.): The Concise Oxford English Dictionary of Current English, 9th edn. Oxford University Press, Oxford (1995)
20. Tsai, W.T., Malek, M., Chen, Y., Bastani, F.: Perspectives on service-oriented computing and service-oriented system engineering. In: Proceedings of the Second IEEE International Symposium on Service-Oriented System Engineering, Washington, DC, USA, pp. 3–10. IEEE Computer Society, Los Alamitos (2006)

A Context-Aware Trust Model for Service-Oriented Multi-Agent Systems*

Kaiyu Wan[1] and Vasu Alagar[2]

[1] Department of Computer Science, East China Normal University, China
kaiyu.wan@gmail.com
[2] Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada and X'ian Jiaotong-Liverpool University, Suzhou, China
alagar@cs.concordia.ca

Abstract. Service-oriented systems offer the potential for minimizing the development time of business applications within an enterprise while promoting collaborative joint ventures among enterprisers distributed geographically. Service-oriented applications assembled from services obtained from different vendors, sometimes anonymous, should be trustworthy. In this paper we investigate context-aware multi-agent systems (MAS) which can dynamically form coalitions of trusted partners as an effective mechanism to act on behalf of service requestors, find services requested by them, determine trusted services, and provide services to the requestors without violating the privacy of the partners involved in such transactions. The MAS is open with respect to external agents requesting and receiving services, but closed with respect to other activities initiated by external agents. The agents in MAS may have different trust models. We explain how trust models of different agents should be composed into a web of trust for trusted transactions in MAS.

1 Introduction

The concept of service-oriented computing system is not new. Its concepts and characteristics have been detailed in many works [4,23,11]. What is new in this paper is the introduction of *context-aware* MAS as a means of discovering and delivering dependable services to its clients in an open distributed environment. Agents in MAS can dynamically form coalitions of trusted partners as an effective mechanism to act on behalf of service requestors, find services requested by them, determine trusted services, and provide services to the requestors without violating the privacy of the partners involved in such transactions. The MAS is open with respect to external agents requesting and receiving services, but closed with respect to other activities initiated by external agents. The trust models of agents, which may be different, can be combined to generate global trust for agent collaboration in delivering services.

Dependability, recently coined as *trustworthiness* [22], is a composite property. It includes *safety*, *security*, *service availability*, and *reliability* attributes. A service is trustworthy if it is reliable, available without interruption, secure, and safe to use. In order

* This research is supported by a Research Grant from Natural Sciences and Engineering Research Council of Canada(NSERC).

G. Feuerlicht and W. Lamersdorf (Eds.): ICSOC 2008, LNCS 5472, pp. 221–236, 2009.

that an agent recognize and evaluate these attributes in a service, we require it to be context-aware. In agent literature, the BDI (belief, desire, intention) semantics is the semantics for agent actions and interactions. We may interpret the full power of BDI as providing the awareness for an agent. Agents indulging in service-oriented activity may be regarded as software agents, who need not have the full power of BDI semantics, instead its *internal* and its *external* awareness constitute a sufficient semantic basis for its actions.

1.1 Awareness and Context

An agent is aware of its internals such as resources under its care, knowledge of its domain of activity, history of its activities and interactions, and norms (policies) that constrains its actions. We call the internal awareness of an agent as *self-awareness*. The external awareness of an agent includes a knowledge of its environment that may include physical devices, humans and other agents. In order that the system response with respect to a request from its environment be equivalent to a cognitive reaction, the "feel" component of external awareness should be factored in. This is realized by introducing the agent to the *world* with which it has to react. This world consists of the information space of its clients and their behavior. The information space is, in general, *multi-dimensional* where each dimension has information or data with respect to one sort. Aggregating dimensions along with information along the dimensions give rise to a formal definition of context [3,28,29], and justifies formalizing external awareness using contexts. External awareness is known as context-awareness.

Context is *rich* concept. Carnap, in his famous treatise [8] *Meaning and Necessity*, distinguished between the *intension* and *extension* of a natural language statement. Intension is the uttered statement and extension is its truth values in different contexts. As an example, the intension of the statement "The physician in charge of the clinic works from 9 am to 3 pm." is itself, and its extensions are evaluated when the physician's name, the clinic's name, and a time are specified. The world of information to evaluate this statement has three dimensions, namely Physician_name, Clinic_name, and Time. A context is constructed when a value/data is given along each dimension. This meaning of context was tacitly understood and first used in AI, and HCI applications. Since then, context has found several applications in a variety of fields of study within computing, however context itself was represented and interpreted only in an ad-hoc manner. Just to give an overview of its spectrum, see [21,7,14] for logic of context and reasoning in AI, [1] for a tutorial on efforts in formalizing context, [9,10] for HCI applications, [16,17,18] for using context within semantic web for service compositions, [3,28] for high-level languages with context as first class objects, [24,27] for context-aware system architecture and [2,30] for context-aware enforcement of privacy, identity management, and trust. The ability to automatically detect contexts, dynamically de-construct and reconstruct contexts are very relevant to context-aware computing, in particular for mobile and pervasive computing applications. Without a formal representation of context, adaptation and reasoning about the consequences of adaptation is not possible. The formal syntax and the semantic domain for context calculus that we have developed in our research are briefly described in Section 2.

1.2 Trust

Trust, similar to context concept, is also a rich concept. It has a pivotal role in human relationships. McKnight and Chervany [19] have given a typology for classifying trusting behavior in domains such as sociology, psychology, political sciences, and management. A trusting behavior, when expressed in natural language and semantically interpreted through Carnap's intensional logic, becomes an intensional statement. The natural language statements corresponding to these trusting behaviors have a natural intensional logic interpretation. This is the rationale behind the intensional trust model discussed in [30]. In Section 3.3 we show that the six-fold trusting behavior discussed in McKnight and Chervany [19] can be cast within context-aware trust model. Examples 1 and 2 illustrate the necessity to integrate context and trust.

Example 1. *It is remarked in [15] that* inaccuracies in the Wikipedia has been rumored to cause students to fail courses. *Apparently this fault is attributed to a "free for all" approach to the contribution of articles by internet users. Anybody can write an article irrespective of the contributing author's expertise in that area. There is no mechanism to verify either the correctness or completeness of the content in an article. A user who accepts the services offered by such sites will eventually lose her trust in the integrity of the service provider.*

Example 2. *This example is taken from [1], and perhaps the starting point for a formal treatment of context by McCarthy [21]. MYCIN [25] is a computer-based medical consultation system. In particular it advises physicians on treating bacterial infections of the blood. When MYCIN is given the information "the patient has* Chlorae Vibrio*" it recommends two weeks of* tetracycline *treatment and nothing else. What it does not reveal is that there is massive* dehydration *during the course of the treatment. While the administration of* tetracycline *would cure the bacteria, the patient would perish long before that due to diarrhea. Here is an instance where context of usage is not explicitly stated. Although the treatment is in principle correct, it is incomplete. The service is totally* unsafe.

If there were to exist a trust model for a system, that model can automatically assign a trust degree based on the completeness and correctness of the information content and offer a personalized rating for every service provided by the system. Completeness and correctness are contextual concepts. Consequently, integrating the trust model with the system awareness of its internal and external contexts, we aim to make the system trustworthy.

1.3 Contributions

We give a rigorous approach to integrate trust model with context-awareness for agents in a MAS. The paper is organized as follows. In Section 2 we briefly outline context formalism. We describe context-aware trust model for service-oriented systems in Section 3. In Section 4 we give a formal context-aware MAS architecture and explain how trusted services are provided by the agents in the absence of a centralized authority. We conclude the paper in Section 5 with a summary of our ongoing research on trustworthy MAS. Appendix 1 lists the formal concepts that we use in MAS architecture.

2 Context Formalism

Context has several working definitions. According to Dey [9] context is *any information that can be used to characterize an entity*, where an entity is *a person, place, or object that is considered relevant to the interaction between user and application*. Context is considered as a *semantic object* by Mrissa etal [18] and this definition is quite close to the formal definition of Wan [29]. Since agents in a MAS need to dynamically construct, assemble, disassemble, modify, and reason with context, a formal definition is necessary. The five most important dimensions from which they should collect information are *who*, *what*, *where*, *when*, and *why*. They respectively provide information on *perception*, to recognize the software component or agent that provides the service or requires the service, *interaction* to determine the type of service, *locality* to determine the location (and its constraints) for providing the service, *timeliness* that specify time bounds for (safe) service delivery, and *purpose* for requested service. It is natural therefore to define a context as a structured typed data: Let $\tau : DIM \rightarrow I$, where $DIM = \{X_1, X_2, \ldots, X_n\}$ is a finite set of dimensions and $I = \{a_1, a_2, \ldots, a_n\}$ is a set of types. The function τ associates a dimension to a type. Let $\tau(X_i) = a_i$, $a_i \in I$. Define a context c as an aggregation of ordered pairs (X_j, v_j), where $X_j \in DIM$, and $v_j \in \tau(X_j)$. The dimensions and the types are suggested by the application of interest.

Example 3. *The vital information to seek an on-line health care service should include the service requestor's credentials. A credential along with location information, date and time can constitute the context. The dimensions are PN (patient name: type string), HC (hospital card number: type integer), LOC (the location where service should be provided: type string), $DATE$ (the date when the service is to be provided: type record).*

2.1 Syntax and Semantics

The context type is determined by the set of dimensions DIM, the types I and the function τ. Our discussion applies to contexts of the same type. The syntax for a context is $[X_i : v_i, \ldots, X_j : v_j]$. The binding between dimensions and the values from the types associated with the dimensions is made explicit. The necessity is that agents should be aware of the dimensions from which the information are perceived. An instance of a context for the type in Example 3 is $[PN : Bob, HC : 123456, LOC : Boston, DATE : \langle 2008/10/30 \rangle]$. For a context c, we write $dim(c)$ to denote the set of dimensions in c, and $val(c)$ to denote the set of typed values in c.

The semantics for contexts is adapted from the set theoretic and relational semantics. we define context operators whose operational semantics are close to the set theoretic and relational operators. Table 1 lists these operators, their precedence and their meanings. By an application of this operational semantics a context expression is evaluated, as in Example 4.

Example 4. *Let $c_1 = [PN : Bob, HC : 123456, LOC : Boston, DATE : \langle 2008/10/30 \rangle]$, $c_2 = [PN : Alice, HC : 456123, LOC : Chicago, DATE : \langle 2008/11/15 \rangle]$, and $c_3 = [PN : Tom, HC : 142536, LOC : Boston, DATE : \langle 2008/10/30 \rangle]$ be three different contexts of the type in Example 3. The expression $c_1 \oplus c_2 \uparrow \{LOC,$*

Table 1. Context Operators and Precedence

operator name	symbol	meaning	precedence
Union	$\sqcup$	Set Union (dimension, value pairs)	3
Intersection	$\sqcap$	Set Intersection (,, ,,)	3
Difference	$\ominus$	Set Difference (,, ,,)	4
Subcontext	$\subseteq$	Subset of dimensions	6
Supcontext	$\supseteq$	Superset of dimensions	6
Override	$\oplus$	Function overwrite	4
Projection	$\downarrow$	Domain Restriction	1
Hiding	$\uparrow$	Range Restriction	1
Undirected Range	$\rightleftharpoons$	Range of simple contexts with same domain	5
Directed Range	$\rightharpoonup$	Range of simple contexts with same domain	5

$DATE\}$ *evaluates to the context* $[PN : Alice, HC : 456123, LOC : Boston, DATE : \langle 2008/10/30\rangle]$. *The result reflects the status wherein* $Alice$ *wants service to be provided at* $Boston$ *on* $2008/10/30$.

Many other properties, such as $dim(c_1 \sqcup c_2) = dim(c_1) \cup dim(c_2)$, can be derived by an application of the semantics[1]. It is sufficient to admit in MAS contexts in which the dimensions are distinct. A context in which a dimension is repeated is equivalent to a set of contexts in each of which the dimensions are distinct [29]. As an example, if Bob wants health care services on two different dates one can construct $c = [PN : Bob, HC : 123456, LOC : Boston, DATE : \{\langle 2008/10/30\rangle, \langle 2009/01/05\rangle\}]$, in which the $DATE$ dimension is repeated. Context c is equivalent to the set with two contexts $c_s = \{[PN : Bob, HC : 123456, LOC : Boston, DATE : \{\langle 2008/10/30\rangle], [PN : Bob, HC : 123456, LOC : Boston, DATE : \{\langle 2009/01/05\rangle\}]$.

Reasoning with Contexts

Let α be a logical expression on some quality attributes that must be verified in a context c. We write $\mathbf{vc}(c, \alpha)$ to denote such a verification condition. The dimensions of a context and the quality attributes that are to be verified in that context may sometime overlap. In that case, the expression α will involve those common dimension names and are treated as free variables. While evaluating α in context c, the dimension names in the logical expression should be bound to some values in their respective typed domains. If sufficient information is not available to evaluate α in context c, the evaluation is postponed, not abandoned. Some simple axioms for verification within a context are the following:

If α is true in a context c_2 then it is true in every sub-context of c_2. That is,

$$\frac{c_1 \subset c_2}{\mathbf{vc}(c_2, \alpha) \Rightarrow \mathbf{vc}(c_1, \alpha)} \tag{1}$$

If α is true in context c then there exists a context c' such that $\mathbf{vc}(c', \mathbf{vc}(c, \alpha))$. That is, for every context c there exists an *outer* context c' from which the verified truths can be observed.

[1] Note that *cup* is a set operator, whereas $\sqcup$ is a context operator.

$$\frac{\mathbf{vc}(c, \alpha)}{\exists c' \bullet \mathbf{vc}(c', \mathbf{vc}(c, \alpha))} \tag{2}$$

Analogous to the Law of Excluded Middle we give the axiom

$$\frac{\mathbf{vc}(c, \alpha \rightarrow \beta), \mathbf{vc}(c, \alpha)}{\mathbf{vc}(c, \beta)} \tag{3}$$

A context c' is a *consistent extension* of context c, written $c' \succeq c$, if $dim(c) \subset dim(c')$, and $\mathbf{vc}(c, \alpha) \Rightarrow \mathbf{vc}(c', \alpha)$. A consistent extension is *monotonic*. Suppose not enough information is available in context c to evaluate α. Since context c' has more (precise) information than context c, $\mathbf{vc}(c', \alpha)$ may have the information to evaluate α.

2.2 Modeling Contact Awareness

A service providing site (agent) must be aware of the different contexts that it is in and the contexts encountered by it. The first kind of awareness is its self-awareness or internal awareness. The second kind of awareness is its external awareness.

Internal Awareness: It is modeled with a view to protect the critical assets of the agent, regulate the service, and optimize resources. It is necessary to abstract in it information regarding it's *clients*, *assets*, *permission policies*, and *obligations*. A client can be an agent acting on behalf of someone, or another entity in the system. In general, they are the subjects who are active in seeking its services. The agent maintains a database (or has access to a database which it can query) of its client categories and identification of clients in each category. Authorization to access the site, get service, and query it can be regulated by identity/password combination or credentials. Assets are the objects that are under its control/ownership. Access to every asset is controlled by the policy (security and privacy) of the institution that the agent represents. For each asset under its control and for each client a permission is determined dynamically, taking into consideration the asset category, the client category, the context of service requirement, and the *purpose* behind the request. An obligation is a policy. There are two kinds of obligations. One kind is "mandatory" that specifies the action (response) that the agent must perform after a user request is fulfilled. For example, if a request for a service is denied, the client shall be informed to get a new authorization, before her session may be terminated. The second kind is obligation set by clients. For example, when a client pays on-line for downloading a software from the site under the agent's control, she may set a time limit for getting a refund on it in case the software fails to work in the configuration set up of the client. Such obligations are usually time constrained. From the above modeling elements, the context type for self-awareness can now be constructed. Assume that $UC = \{UC_1, \ldots, UC_m\}$ is the set of client categories as determined by the site. Let $AC = \{AC_1, \ldots, AC_k\}$ be the set of asset categories which are to be protected. We regard UC_i's and AC_js as dimensions for this context type. Let PC denote the purpose dimension. Assume that the type for UC_i is *string*, the type for AC_i is the set of integers (pointers to files), and the type for PC is the set $\{\text{Accounts}, \text{Clinical}, \text{Registry}\}$. An example of context in this type is $[UC_1 : Alice, AC_2 : EI_1, PC : \text{Clinical}]$. In this context $Alice$ in user category UC_1 is requesting access to the asset EI_1 in category

AC_2 required for a clinic. An obligation is expressed as a logical expression. An example of obligation is $\alpha = onduty(Alice) \wedge 2 \leq deliver\_clinic(Metabolism) \leq 5$, to be interpreted as that $Alice$ is on duty and she must deliver the asset within the time limit $[2, 5]$. In order to enforce the obligation the verification condition $\mathbf{vc}(c, \alpha)$ must be fired at context c.

External Awareness: Contexts that characterize external awareness are constructed automatically from the information gathered from client profile, and client request. The relevant dimensions are LOC, $TIME$, WHO, $WHAT$, $WHERE$, $WHEN$, WHY which correspond respectively to the location from where service is requested, the date/time at which the service request is made, the client role, the nature of service, the location where the service should be provided, the date/time by which the service should be given, and the reason for requesting the service. An example of external context is $c = [LOC : \text{Montreal}, TIME : d_1, WHO : Manager, WHAT : \text{EPR2}, WHEN : d_2, WHY : \text{Accounts}, WHERE : \text{Boston}]$.

3 An Overview of Trust Model for Service-Oriented Systems

The MAS model that we discuss in Section 4 differs from peer-to-peer servicing systems. The MAS has a middle layer in which agents act as mediators between service requestors and service providers. Service requestors should communicate the quality attributes to the mediators, rather than absolute trust values they expect for a service. The mediators discover the service that satisfies the quality attributes and deliver the service together with a trust value (in a standard ordinal scale) associated with the service to the service requestor. The advantages of this approach include ensuring privacy of service requestors, avoiding the conflicting views on absolute trust values coming from different sources, and guaranteeing the quality of service on behalf of the many service providers whose services might have been composed into the delivered service. We believe that it is necessary to separate trust calculation from trust verification for the following reasons.

Every trust model, regardless of the mechanisms used to measure trust and types of values used to assess and compare trust, will use (1) a trust function, which assigns a trust value on an entity in the context of service request, and (2) enforce trust policies for trust evaluation, and propagation. A system or service is defined to be trustworthy [22] if the four quality attributes *safety*, *security*, *reliability*, and *availability* can be verified in it. If we could formulate these attributes into a logical expression α, then we have to verify $\mathbf{vc}(c, \alpha)$, in every context in which service is provided. This implies that the dimensions used in constructing the service providing context must be sufficiently expressive to include these four quality attributes. The number of dimensions may dramatically increase, and in practice it is not easy to identify all dimensions. Hence, trust calculation, which may be only approximate, must be separated from verification which should be exact. In order to make verification feasible, only those parameters that are recognized as most important for a specific application should be included in the verification condition. As an example, in a movie downloading site the ratings is the most important parameter and it must be verified in the service.

3.1 Choice of Trust Domain

In the literature there exists different conventions for choosing trust values. Regardless of the choice, the trust domain $\mathcal{D}$ which includes all the trust values for an application should be a *complete partial order* [30]. Integers, and real numbers are the natural choices that fit this requirement. If symbols are used, as in stock recommendations or in grading, then we regard it as an *enumerated* type, the order being implicit in the enumeration. As an example, in the enumeration $\{hold, sell\}$, the degree of trust in $sell$ recommendation is higher than the degree of trust in $hold$ recommendation. When the trust domain is either non-numeric or non-simple (such as vectors), it is necessary to define a *metric* on the trust domain to enable the measurement of the distance between two trust values. Such a measurement helps to understand the disparity or closeness between trust values. Towards such a definition of metric here is a simple approach. If $\sigma : \mathcal{D} \rightarrow \omega$ is a total monotone function and $\simeq$ is a partial order on the trust domain then for $d_1, d_2 \in \mathcal{D}$ (1) if $d_1 \simeq d_2$ then $\sigma(d_1) \leq \sigma(d_2)$, and (2) for every chain $d_1, \ldots, d_k$ in $\mathcal{D}$, $\sigma(d_1) \leq \sigma(d_2) \leq \ldots \sigma(d_k)$. This way of metricizing the trust domain is an abstraction of the way that PICS rankings [20,13] are usually interpreted.

3.2 Context-Aware Trust Measurement

We denote the set of entities (subjects and objects in the system) by $\mathcal{E}$, the set of contexts by $\mathcal{C}$, and the set of logical expressions over quality attributes by $\mathcal{Q}$. For some $\alpha \in \mathcal{Q}$, if $\mathbf{vc}(\mathbf{c}, \alpha)$ is true then the trust value in context c should be based upon the quality attributes in α and the information available in context c.

Definition 1. *The function,*

$$\pi : \mathcal{E} \times \mathcal{E} \times \mathcal{C} \times \mathcal{Q} \rightarrow \mathcal{D}$$

associates for $a, b \in \mathcal{E}$, and $c \in \mathcal{C}$ a unique element $d \in \mathcal{D}$, called the trust that a has on b in context c, provided $\mathbf{vc}(\mathbf{c}, \alpha)$ is true.

A service-oriented system is an open distributed system, where sites may have different trust domains and context types. In order to interpret the trust value in one context of one site in a context of another site, a mapping between their trust domains, and another mapping between their context types must be provided. As an example, let us fix the context as common for both sites, say on-line auction, and one site is in Euro zone and another site is in US dollar zone. The sites must have a type conversion function to interpret one another's bidding. In a decentralized MAS, as we explain in Section 4, both type conversions and context homomorphisms can be done by agents.

Reasoning with Context-aware Trust
We fix the context type and trust domain, and give rules for reasoning with trust values.

$$\frac{\mathbf{vc}(\mathbf{c}, \alpha) \wedge \mathbf{vc}(\mathbf{c}, \beta)}{\pi(a, b, c, \alpha \wedge \beta) = \text{maximum}\{\pi(a, b, c, \alpha), \pi(a, b, c, \beta)\}} \tag{4}$$

$$\frac{\mathbf{vc}(\mathbf{c}, \alpha) \wedge \alpha \Rightarrow \beta}{\pi(a, b, c, \beta) = \pi(a, b, c, \alpha)} \tag{5}$$

Corresponding to the axiom 2 we postulate

$$\frac{\exists\beta \bullet \mathbf{vc}(\mathbf{c}', \beta) \wedge \beta \Rightarrow \alpha}{\pi(a, b, c, \alpha) = \pi(a, b, c', \beta)} \tag{6}$$

That is, what is observed from c' should not be invalidated at c'. Corresponding to the subset axiom that if $c_1 \subset c_2$ then $\mathbf{vc}(c_2, \alpha) \Rightarrow \mathbf{vc}(c_1, \alpha)$, we postulate

$$\frac{\mathbf{vc}(\mathbf{c_2}, \alpha) \wedge \mathbf{c_1} \subset \mathbf{c_2}}{\kappa(\pi(a, b, c_1, \alpha) = \pi(a, b, c_2, \alpha))} \tag{7}$$

where κ is the *belief* operator. The trust value, that is believed at one context, can be changed in the same context when more constraints are added to α. In the following equation $\mathbf{vc}(\mathbf{c_1}, \alpha) \wedge \mathbf{vc}(\mathbf{c_2}, \beta)$ is true.

$$\pi(a, b, c_1, \alpha) \geq (\pi(a, b, c_1 \sqcap c_2, \alpha \wedge \beta) \tag{8}$$

$$\pi(a, b, c_2, \beta) \geq \pi(a, b, c_1 \sqcap c_2, \alpha \wedge \beta) \tag{9}$$

$$\text{lub}\{\pi(a, b, c_1, \alpha), \pi(a, b, c_2, \beta)\} \geq \pi(a, b, c_1 \sqcap c_2, \alpha \wedge \beta) \tag{10}$$

where the symbol lub is the *least upper bound* operator. In the next equation assume that $\mathbf{vc}(\mathbf{c_1}, \alpha) \vee \mathbf{vc}(\mathbf{c_2}, \beta)$ is true.

$$\pi(a, b, c_1 \sqcup c_2, \alpha \vee \beta) \geq \pi(a, b, c_1, \alpha) \tag{11}$$

$$\pi(a, b, c_1 \sqcup c_2, \alpha \vee \beta) \geq \pi(a, b, c_2, \beta) \tag{12}$$

$$\pi(a, b, c_1 \sqcup c_2, \alpha \vee \beta) \geq \text{glb}\{\pi(a, b, c_1, \alpha), \pi(a, b, c_2, \beta)\} \tag{13}$$

where the symbol glb is the *greatest lower bound* operator.

3.3 Trust Model vs. Trusting Behavior

We illustrate that the six-fold classification of trusting behavior of an entity a, called *trustor*, on another entity b, called *trustee*, given by McKnight and Chervany [19], can be expressed within context-aware trust model. A trusting behavior can also be associated with one or more of the trustworthiness features *security*, *safety*, *reliability*, and *availability*.

1. *Disposition:* An entity a (client) is naturally inclined to trust b (a vendor). The intention affects directly the actions, without going through any reasoning process. That is, a trusts in *her judgement* in order to trust b, which implies that her judgement on attributes expressed in the expression α is true in every context c the service is offered by b.
2. *Situation:* An entity a trusts b in a particular scenario. As an example, a (consumer) may trust b (broker) in the context c_1 of buying mutual funds but not in the context c_2 of buying stocks. That is, for entity a $\mathbf{vc}(\mathbf{c_1}, \alpha)$ is true and $\mathbf{vc}(\mathbf{c_2}, \beta)$ is false where α is the assertion on the quality attributes for buying mutual funds and β is the assertion on the quality attributes for buying stocks. Axioms 8-13 hold.

3. *Structure:* An entity a trusts the structure (institution) of which the entity b is a member. For example, if b is a chartered bank and a knows the norms of the federal bank of which all chartered banks are affiliates a has the natural inclination to trust b. If b is the mortgage officer in a chartered bank and a is a customer who understands the bank policies on privacy, investments, and mortgage loans, a is inclined to trust b. All the axioms enumerated above are valid. Most importantly trust is *transitive* because of the axioms 3, 5-6.
4. *Belief:* An entity a believes that b is trustworthy. The belief may be based upon factors such as *predictable behavior*, *timeliness*, and *integrity of service*. Consumer a trusts site b because whenever she downloads a software the downloaded software behaves as specified. Trust is calculated in a context, based upon the trust values in the history of contexts encountered by the entity. The belief axiom 7 can be generalized so that trusts at two different contexts, not necessarily subsets, are comparable.
$$\frac{\mathbf{vc}(\mathbf{c_1}, \alpha) \wedge \mathbf{vc}(\mathbf{c_2}, \beta) \wedge \alpha \Rightarrow \beta}{\pi(a, b, c_2, \beta) = \pi(a, b, c_1, \alpha)} \tag{14}$$
5. *Behavior:* An entity a voluntarily depends on entity b. There is no third party involvement here. Consumer a trusts the services provided by a site b because she is either unable to find that service from another site or has not heard anything bad about b's service. Hence consumer a accepts $\mathbf{vc}(\mathbf{c}, \alpha)$ to be true, even when she does not have sufficient information to actually evaluate it.
6. *Intention:* Intention is the result of a desire to reach a goal in a specific context. Once the goal is set in a context an entity a is willing to depend on the services provided by entity b in order to reach her goal. In order to reach the goal, a will try to extend the current context in a consistent manner at each step. Consumer a may choose different b's at different contexts in order to achieve her goal. Formally, let $c_1 \ldots c_n$ be a sequence of contexts such that c_i is a consistent extension of c_{i-1}, $\mathbf{vc}(\mathbf{c_i}, \alpha_\mathbf{i})$ is true, and $\alpha_n \Rightarrow \alpha_{n-1} \ldots \alpha_2 \Rightarrow \alpha_1$. Then we can deduce from the axioms that $pi(a, b, c_1, \alpha_1) \leq pi(a, b, c_2, \alpha_2) \leq \ldots \leq pi(a, b, c_{n-1}, \alpha_{n-1}) \leq pi(a, b, c_n, \alpha_n)$. That is, until the goal is reached the trust values are monotonic non-decreasing, implying that no contradiction is ever introduced.

4 MAS Model of Service

A service delivered in a context c is a function [6] which should satisfy the quality attributes α if $\mathbf{vc}(\mathbf{c}, \alpha)$ is true. To receive and process a service request we model the MAS with four types of agents. The types are defined by the roles and the normative behavior of agents. Many agents of one type may be active in each session of service processing. We write $x : X$ to mean that agent x is of type X.

4.1 Agent Types

The four agent types in the system are UIA, MMA, SPA, and TSA. An agent of type UIA assists users external to the MAS, in formulating service requests, compiling their profiles, deducing the user's information needs by direct communication and history

of transactions, presenting the requests to agents of type MMA, receiving response to a request from MMA agents and communicating them to the respective users, taking corrective actions on behalf of the user, and adapting to the changes in the environment so that it can improve its assistance to the user. An agent of type MMAinteracts with UIA agents to receive service requests and return services, interacts with SPA agents to discover and receive services, and interacts with TSA agents for authentication, certification, trust and context management. An agent of this type SPA provides services in one specific application domain. As an example, *Air Travel* is one specific application within *Transportation* domain. In the MAS model, an SPA agent will provide service in one such specific application domain. Agents of type TSA act as certification authorities, context managers, trust managers, configuration managers, ontologists, vigilante, and auditors. While acting as a vigilante it will monitor events, maintain a history of system evolution, draw inferences, and plan actions to ensure that no agent from outside the AMS has intruded into the system. An important requirement is that all agents in MAS trust the agents who manage their trust information in a secure and manner, assuring integrity and confidentiality. The TSA agents are assumed to have been given sufficient resources to ensure that the services are provided without getting interrupted by external attacks and internal failures. More than one agent may act out the same role, however an agent may act only in conformance with the norms prescribed for its type. An agent may advertise its roles to other agents in MAS and may *subscribe* for their services. An agent may deny service to an agent not subscribed to its services. We use the notation $x : X$ to denote that x is an agent of type X.

4.2 Service Protocol

A service interaction between two agents can happen only if at least one of them is subscribed to the services published by the other. An external client of MAS can subscribe to one or more agents of type UIA. No other MAS agent is visible to an client. Although an external client may subscribe to more than one UIA agent, she cannot interact with more than one UIA agent during a session. A session of a client u, denoted u_s, starts from the instant u contacts one of her UIA agents, say $u_{uia} : UIA$, for service. The session u_s ends when u_{uia} either delivers the service to u or informs u that the requested service is not deliverable. The assumption is that all external clients subscribe prior to starting their sessions.

A subscription is made to an agent when a profile of the requestor is registered with it. The profile includes a specific set of services and a verifiable credential of the requestor. The attributes that make up a profile are determined by the agent to whom the subscription is made. A submitted profile may be modified by the subscriber after the first submission. In turn the agent is obliged to ensure the confidentiality of the registered profiles by announcing the trusted authority in the system with whom the profiles are registered. That is, only a trusted authority agent $taa : TSA$ can maintain the database of profiles collected by an agent u_{uia}. More than one UIA agent may use the services of a taa agent. However, an agent u_{uia} cannot use the services of more than one taa agent.

Protocol Steps

1. *submit service request:* User u contacts one of the agents of type UIA. This micro-step can be just a handshake based on password authorization. Agent u_{uia} from this set presents u with a service template. The user returns an instance of the service template, in which information that can identify the user, a service (domain of service, functionality of service), quality attributes for the service, and contextual information for service delivery are specified. This instance is the specification of the service request, called S_u. The agent u_{uia} sends S_u to its trusted authority taa_{uia}, to the context-management agent $cma : TSA$, and to the matchmaking agent $mma : MMA$[2]. The service specification S_u is structured in such a way each agent that receives S_u can understand and decide which part of S_u is relevant for its task.
2. *matching a profile:* Agent taa_{uia} ignores the information in S_u that is not relevant to its task, and uses the rest of the information to match the profiles in its database. It selects a set of profiles from its database[3] such that in every selected profile the service specified in S_u matches exactly, and the identification information in S_u is either completely or partially matched. If the set of selected profiles is empty it informs u_{uia}, otherwise it sends the selected profile to the authentication agent asa.
3. *authentication of client:* Agent asa has sufficient knowledge in its knowledge base to construct logical assertions from submitted information and reason whether or not a submitted information implies already subscribed information. That is, from the credentials in S_u it forms a logical expression α, from the matching profile of the client received from taa_{uia} it forms another logical expression β and will evaluate $\beta \Rightarrow \alpha$. If it is true, then from the fact that β it can conclude α is true and authenticate the client. It includes the result of authentication in S_u and sends it to u_{uia} and mma.
4. *context construction:* The agent cma has an implementation of context calculus. It extracts the contextual information c_u at which request for service is made and the contextual information c'_u for service delivery from S_u. Context c'_u includes quality attributes. It constructs the contexts in the syntax described in Section 2. It includes the contexts in S_u and send it to mma.
5. *notification of authentication failure:* If u_{uia} receives back S_u a notice of authentication failure, it informs the user u and the security agent asa.
6. *matchmaking process:* Agent mma receives from cma contexts c_u and c'_u. Context c_u is the context in which client u has requested the service, and context c'_u includes the constraints on the quality and other non-functional constraints (such as hardware) to be respected at service delivery to the client. Agent mma has a table of service information gathered from the service providers. For each service provider spa there is an entry in the table, showing the names of services provided by spa, the functionality and quality attribute for each service, and obligation constraints, if any, for each service. We defer a formal discussion on these issues, and assume that mma has the knowledge and resources to match a request against the

[2] For simplicity of exposition assume that there exists only one context management agent, one authentication/security agent, and one match making agent in the system.

[3] The database is a shared resource between trusted authority agent and authentication agent.

services in the table, choose services, and compose them to meet the request. The context-based Service composition approach discussed by Mrissa et al; [18] can be the basis of the knowledge-base of mma. What is needed is to extend this approach so that mma can verify the satisfaction of the composition to the service delivery context c'_u. A brief description of these steps follow. We let $spa_1, \ldots, spa_k$ be the service providers whose services are needed for the composition.

checking credentials of client: Agent mma evaluates $\mathbf{vc}(\mathbf{c_u}, \mathbf{dc_i})$, where dc_i is a logical expression composed from the institutional policy governing service provision by spa_i. A service provision agent represents some institution and is bound by the policy of that institution. A policy is rule, which being declarative, can be formulated into a logical expression. As an example, the hospital policy regarding physicians to access patient records from the nursing home and for the purpose specified in the context c_u, may have to be evaluated at this stage. In order that u may be permitted to receive the service the expression $\bigwedge_i\{\mathbf{vc}(\mathbf{c_u}, \mathbf{dc_i})\}$ must be true. If there exists a service provider spa_j for whom $\mathbf{vc}(\mathbf{c_u}, \mathbf{dc_i})$ is false, then that service provider is removed from the list of service providers, and agent mma will contact another spa to get an equivalent service.

service composition: After a successful termination of the previous step, services are composed. Let f denote the composition of services. If α is the logical expression conjoined from the quality attributes (security, trust, reliability, availability, performance) of all spas whose services are composed into f, the assertion claimed by the service providers is that f *satisfies* α (denote as $f\ sat\ \alpha$). That is, the property α is verifiable in the function f. According to Hoare logic, it is sufficient to verify that the post-condition of f implies α.

conformance with expected quality: Agent mma evaluates $\mathbf{vc}(\mathbf{c'_u}, \alpha)$. If it is true, then because the agent has already proved $f\ sat\ \alpha$) it can conclude that α conforms to the quality attributes specified in the service delivery context c', and consequently f is acceptable in context c'_u.

trust calculation: Agent mma sends the list of service providers, the context c'_u and α to the trust management agent tma, which calculates a trust value for mma on each spa_i. The agent tma applies homomorphism and type conversion to normalize the trust values to an ordinal scale, and returns the resulting trust values $\pi(mma, spa_i, c'_u, \alpha)$ to mma.

obligation calculation: Agent mma calculates $\beta = \bigvee_i\{\mathbf{vc}(\mathbf{c'_u}, \mathbf{ob_i})\}$, the obligation constraint.

service provision: From the trust values received, agent mma computes one trust value[4], and sends the service f, the obligation constraint β, along with the computed trust value to agent u_{uia}.

7. *service delivery:* Agent u_{uia} delivers the service f to u, informs u the degree of trust in the service, and enforces the obligation β on behalf of the client u. The enforcement is a firing of the verification condition $\mathbf{vc}(\mathbf{c''_u}, \beta)$, where c''_u is a consistent extension of c'_u.

[4] This can be an *infimum* reflecting the most pessimistic value, or *supremum*, reflecting the most optimistic value, or the average, reflecting a statistical value.

5 Conclusion

In this paper we have given a comprehensive overview of a context-aware trust model for service oriented systems. A service oriented system is envisaged through a multi-agent system, modeled with four agent types. Agents in the MAS can form dynamic coalitions of trusted partners, acting on behalf of service requestors and service providers, and facilitate trustworthy service delivery. The MAS architecture ensures the privacy of participants in a transaction without compromising on the quality of service.

The basic function that computes trust is context-specific, but the actual method used in assessing it is left undefined. We believe that trust formation is a separate issue, and is actively studied by various research groups. Those methods are only empirical. Our approach abstracts the properties of trust, regardless of how they are assessed, and provides a basis for reasoning about trust within a context as well as across different contexts. The important benefits in our approach are: (1) Context is independent of what it references. As a consequence, context-based trust definition captures different kinds information conveyed by events that are observable by agents; (2) Context calculus enables the construction of new contexts from existing contexts, and the logic of contexts, once defined, enables one to reason about the information in the newly constructed context with respect to the information contents in the contexts from which the new one is constructed. As a consequence context-based trust definition is well-suited to handle trust in dynamic networks, in which contexts and their respective information contents may dynamically change independent of each other. Our ongoing work includes a prototype implementation to evaluate the impact of the model on performance attributes, development of a case study on which the theory and performance evaluation are to be applied, and a thorough comparison between our approach and the approach carried out in the semantic web research forums.

References

1. Akman, V., Surav, M.: Steps toward Formalizing Context. AI Magazine, 55–72 (Fall 1996)
2. Alagar, V., Wan, K.: Context Based Enforcement of Authorization for Privacy and Security in Identity Management. In: de Leeuw, E., Fischer Hubner, S., Tseng, J.C., Borking, J. (eds.) IFIP International Federation for Information Processing. Springer Series, vol. 261, pp. 25–38 (2008)
3. Alagar, V.S., Paquet, J., Wan, K.: Intensional programming for agent communication. In: Leite, J., Omicini, A., Torroni, P., Yolum, p. (eds.) DALT 2004. LNCS, vol. 3476, pp. 239–255. Springer, Heidelberg (2005)
4. Barry, D.K.: Web Services and Service-Oriented Architecture: The Savvy Manager's Guide. Morgan Kaufmann Publishers, San Francisco (2003)
5. Bucur, O., Beaune, P., Boissier, O.: Representing Context in an Agent Architecture for Context-Based Decision Making. In: Proceedings of CRR 2005 Workshop on Context Representation and Reasoning (2005)
6. Broy, M.: A Formal Model of Services. ACM Transactions Software Engineering and Methodology 16(1), 1–40 (2007)
7. Buvač, S., Buvač, V.: Mathematics of context. Fundamenta Informaticae 23(3), 263–301 (1995)

8. Carnap, R.: Meaning and Necessity: A study in semantics and modal logic (1947); reprinted by University of Chicago Press (1988)
9. Dey, A.K., Salber, D., Abowd, G.D.: A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-aware Applications. Anchor article of a special issue on Human Computer Interaction 16 (2001)
10. Dey, A.K.: Understanding and Using Context. Personal and Ubiquitous Computing Journal 5(1), 4–7 (2001)
11. Dijkman, R., Dumas, M.: Service-oriented Design: A multi-viewpoint approach. International Journal on Cooperative Information Systems 13(14) (2004)
12. Finin, T., Fritzson, R., McKay, D., McEntire, R.: KQML as an Agent Communication Language. In: Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM 1994). ACM Press, New York (1994)
13. Grandison, T., Sloman, M.: A Survey of Trust in Internet Applications. IEEE Communications Surveys, 2–16 (2000)
14. Guha, R.V.: Contexts: A Formalization and Some Applications, Ph.d thesis, Stanford University (1995)
15. Korsgaard, T.R., Jensen, C.D.: Reengineering the Wikipedia for Reputation. In: 4th International Workshop on Trust Management (STM 2008), Trondeim, Norway, July 16-17, 2008. Electronic Notes in Theoretical Computer Science (2008), www.elsevier.nl/locate/entcs
16. Maamar, Z., Mostefaoui, S.K., Yahyaoui, H.: oward an Agent-Based and Context-Oriented Approach for Web Services Composition. IEEE Transactions on Knowledge and Data Engineering 17(5), 686–697 (2005)
17. Maamar, Z., Benslimane, D., Narendra, N.C.: What can CONTEXT do for WEB SERVICES? Communications of the ACM 49(12), 98–103 (2006)
18. Mrissa, M., Ghedira, C., Benslimane, D., Maamar, Z., Rosenberg, F., Dustdar, S.: A Context-Based Mediation Approach to Compose Semantic Web Services. ACM Transactions on Internet Technology 8(1), 1–23 (2007)
19. McKnight, D.H., Chervany, N.L.: Trust and distrust definitions: One bite at a time. In: Falcone, R., Singh, M., Tan, Y.-H. (eds.) AA-WS 2000. LNCS (LNAI), vol. 2246, pp. 27–54. Springer, Heidelberg (2001)
20. Miller, J., Resnik, P., Singer, D.: PICS Rating Services and Rating Systems (and Their Machine Readable Descriptions) version 1.1, http://www.w3.org/TR/REC-PICS-services
21. McCarthy, J.: Some expert systems need common sense. In: Pagels, H. (ed.) Computer Culture: The Scientific Intellectual, and Social Impact of Computer. Annals of the New York Academy of Sciences, vol. 426 (1984)
22. Mundie, C., de Vries, P., Haynes, P., Corwine, M.: Trustworthy Computing - Microsoft White Paper. Microsoft Corporation (October 2002)
23. Papazoglou, M.P.: Service-oriented Computing: Concepts, characteristics, and directions. In: Proceedings of the Fourth International Conference on Web Information Systems Engineering, Whasington, DC, USA, p. 3. IEEE Computer Society Press, Los Alamitos (2003)
24. Rey, G., Coutaz, J.: The Contextor Infrastructure for Context-Aware Computing. In: Proceedings of 18th ECOOP 2004 Workshop on Component-oriented approach to context-aware systems (2004)
25. Shortcliffe, E.: MYCIN: Computer-based Medical Consultations. Elsivier, New York
26. Toivonen, S., Lenzini, G., Uusitalo, I.: Context-aware Trust Evaluation Functions for Dynamic Reconfigurable Systems. In: Proceedings of WWW 2006, Edinburgh, UK (May 2006)
27. Wan, K., Alagar, V., Paquet, J.: An Architecture for Developing Context-Aware Systems. In: Tzschach, H., Walter, H.K.-G., Waldschmidt, H. (eds.) GI-TCS 1977. LNCS, vol. 48, pp. 48–61. Springer, Heidelberg (1977)

28. Wan, K., Alagar, V.S.: An Intensional Programming Approach to Multi-agent Coordination in a Distributed Network of Agents. In: Baldoni, M., Endriss, U., Omicini, A., Torroni, P. (eds.) DALT 2005. LNCS, vol. 3904, pp. 205–222. Springer, Heidelberg (2006)
29. Wan, K.: Lucx: Lucid Enriched with Context. Ph.d Thesis, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada (January 2006)
30. Wan, K., Alagar, V.: An Intensional Functional Model of Trust. Trust Management II. In: Karabulut, Y., Mitchel, J., Hermann, P., Jensen, C.D. (eds.) IFIP 2008, pp. 69–85. Springer, Heidelberg (2008)

Appendix – Type Conversion and Homomorphism

Homomorphism: In a distributed MAS, agents are most likely to have different trust domains. In order that trust values across agents can be compared and global trust computed it is necessary that a homomorphism ϕ, defined below, be a continuous function on the partial order trust chains.

$$\phi : (\mathcal{D}, \simeq, \sigma) \rightarrow (\mathcal{D}', \simeq', \sigma')$$

- for $d_1, d_2 \in \mathcal{D}$, $\phi(d_1 \simeq d_2) = \phi(d_1) \simeq' \phi(d_2)$
- $\phi(\sigma(d_1, d_2)) = \sigma'(\phi(d_1), \phi(d_2))$

If a homomorphism, as defined above exists, then trust value from one domain can be converted to a trust value in another domain. Let $\pi : \mathcal{E} \times \mathcal{E} \times \mathcal{C} \rightarrow \mathcal{D}$ and $\pi' : \mathcal{E} \times \mathcal{E} \times \mathcal{C} \rightarrow \mathcal{D}'$ be two functions that give the trust values on the trust domains $\mathcal{D}$ and $\mathcal{D}'$. Then $\phi \circ \pi = \pi'$.

Context Type Conversion:

[DIM and I are same:] Let $\tau : DIM \rightarrow I$ and $\tau' : DIM \rightarrow I$ be two context types, $\tau \neq \tau'$. Let $DIM_1 \subset DIM$, and $I_1 \subset I$, such that $\tau = \tau'$ when restricted to $DIM_1 \rightarrow I_1$. Let $DIM_2 = DIM \setminus DIM_1$, and $I_2 = I \setminus I_1$. Then for every $X_i \in DIM_2$, $\tau(X_i) = a_i$, $\tau'(X_i) = a_j$, $a_i, a_j \in I_2$, and $a_i \neq a_j$. However, if the two types a_i and a_j have a common super-type b_{ij}, in the sense defined below, then we can replace values of types a_i and a_j with a value in b_{ij}.

Definition 2. *If there exists a type b_{ij} and two maps $\iota_i : a_j \rightarrow b_{ij}$ and $\iota_j : a_i \rightarrow b_{ij}$ such that $\iota_j \circ \tau = \iota_i \circ \tau'$ then the type b_{ij} can be used for the dimension X_i. Contexts $c_1 = [X_i : x]$, $x \in a_i$ and $c_2 = [X_i : y]$, $y \in a_j$ are type convertible to $[X_i, z]$ because there exists $z \in b_{ij}$ such that $z = \iota_j(x) = \iota_i(y)$.*

[(DIM,I), (DIM,I') are two different pairs:] The context types are $\tau : DIM \rightarrow I$, and $\tau' : DIM \rightarrow I'$. If there exists a map $\phi : I \rightarrow I'$ such that $\phi \circ \tau = \tau'$ then contexts of type τ can be converted to contexts of type τ'.

[DIM and DIM' are two different sets of dimensions:] Let $\tau : DIM \rightarrow I$ and $\tau' : DIM' \rightarrow I'$ be two context types. If there exists $D \subset DIM$ and $D' \subset DIM'$ such that $\mid D \mid = \mid D' \mid$, and for each $d \in D$ there exists a unique $d' \in D'$ such that $\tau(d) = \tau'(d')$ then the dimension pairs (d, d') are alias to each other. Hence, from the context type τ we can project out the sub-contexts on dimensions in the set D, and then rename every context in $d \in D$ with its alias in D'. In practice, this is much easier if the ontologist, an agent in MAS, has the alias table and helps the cma agents to extract contexts that suit the dimensions gathered by its clients.

Three Common Mistakes in Modeling and Analysis of QoS of Service-Oriented Systems

Vladimir Tosic[1,2,3]

[1] NICTA*
[2] University of New South Wales, Sydney, New South Wales, Australia
[3] University of Western Ontario, London, Ontario, Canada
vladat@computer.org

Abstract. Since the basic Web service technologies did not address QoS issues, a huge body of academic and industrial works studied various aspects of QoS modeling/specification, analysis, monitoring, and control for service-oriented systems. Unfortunately, some of these works adopted oversimplifications that are often not appropriate for the complex reality. In this discussion session, we will point out to three such recurring oversimplifications: 1) specifying provider's QoS guarantees without limiting the number of requests; 2) using past QoS measurements to predict future QoS without taking into consideration context of requests; and 3) predicting response time of a sequence of services as a simple addition of their response times without discussing circumstances under which such calculation is valid/invalid. A significant number of authors (often independently from each other) used these oversimplifications without informing readers about their limitations and consequences. We will discuss why such oversimplifications are mistakes and "anti-patterns" in QoS modeling and analysis for service-oriented systems.

Keywords: Quality of service, contract, performance analysis, performance prediction, Web service selection, Web service composition, anti-pattern.

* NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centres of Excellence program.

G. Feuerlicht and W. Lamersdorf (Eds.): ICSOC 2008, LNCS 5472, p. 237, 2009.

Enabling Service Business Ecosystems (ESBE 2008)

Introduction: First International Workshop on Enabling Service Business Ecosystems (ESBE 2008)

This part of this volume contains the proceedings of the **First International Workshop on Enabling Service Business Ecosystems** (**ESBE'08**), held on December 01, 2008 in Sydney, Australia, in conjunction with the International Conference on Service Oriented Computing (ICSOC) 2008.

Today, services are used as a core component or utility of business operations and offer programmatic interfaces to applications to exploit these services. The majority of attention on service oriented systems has been contemplated on its related technical standards and technology integration. However, many of today's available services are not considered as providing relevant business value as their use by third-party clients have unclear terms and conditions with unknown risk. The trend of software transforming to the service oriented paradigm demands a new way of business model reassurance to manage services operation, deployment, and longevity in the context of business ecosystems.

The **ESBE** workshop aims to bring together researchers and practitioners in services development across business domains. Its focus is on creating business value through services and, looking beyond individual businesses, fostering the growth of a service ecosystem.

This year 6 papers were selected in two categories, 4 full research papers and 2 short research papers, based on a thorough review process, in which each paper was reviewed by at least 3 experts in the field. The selected papers illustrate very high caliber research in the field of service ecosystem.

We would like to take this opportunity to express our thanks to all people who have contributed to **ESBE'08**. We are indebted to all members of the program committee for their reviews and comments. We appreciate the opportunity given by the ICSOC workshop chairs to organize this workshop in conjunction with ICSOC 2008.

We hope you find the papers in this proceedings interesting and stimulating.

Vincenzo D'Andrea
G.R. Gangadharan
Renato Iannella
Michael Weiss

G. Feuerlicht and W. Lamersdorf (Eds.): ICSOC 2008, LNCS 5472, p. 241, 2009.

Describing Services for Service Ecosystems

Gregor Scheithauer[1,2], Stefan Augustin[1], and Guido Wirtz[2]

[1] Siemens AG, Corporate Technology, Knowledge Management
Otto-Hahn-Ring 6, 81739 Munich, Germany
[2] University of Bamberg, Distributed and Mobile Systems Group
Feldkirchenstrasse 21, 96047 Bamberg, Germany

Abstract. Service ecosystems are electronic market places and emerge as a result of the shift toward service economies. The aim of service ecosystems is to trade services over the internet. There are still obstacles that impede this new form of market places. Two of these challenges are addressed in this paper: (1) identification of appropriate service properties to specify service descriptions, and (2) a need of a clear classification for service description notations. Therefore, service properties and their relationship are introduced and an adaption for the Zachman Framework is presented to classify service description notations depending on the relative perspective.

Keywords: Service description, Zachman Framework, service ecosystems.

1 Introduction

Tertiarisation describes a structural change in developed countries concerning the sectoral composition. Countries shift from an industry economy toward a service economy. Sources of this change include globalization, technological change, and an increasing demand for services [21]. Considering this trend, it becomes clear that services and the service economy play an important role in today's and tomorrow's business. In line with this trend, service ecosystems [4] emerge, such as eBay, Google Base, Amazon.com, SalesForce.com, and SAP Business by Design. Such market places allow to trade services between different legal bodies.

One major challenge for service ecosystems is the fact that services are different to goods. According to Booms and Bitner [5] services are intangible, and thus, can neither be stored, transported, nor resold. Goods are produced at some point, *stored*, and eventually consumed. In contrast, production and consumption of services take place at the same time. Goods can be *transported* from one point to another. Services, on the other hand, are consumed at customers' location, thus, production and consumption happen in one place. Whereas goods can be *resold*, services' outcome cannot be sold to another party. Additionally, services can hardly be *standardized*, since service experience is unique and depends on individual expectations. Moreover, no established language exists to define, agreeing on, and to monitor service properties [12]. For service ecosystems, service descriptions abstract from concrete services and provide a tangible artifact,

G. Feuerlicht and W. Lamersdorf (Eds.): ICSOC 2008, LNCS 5472, pp. 242–255, 2009.

which can be stored and transported, and therefore relax some of Booms and Bitner's arguments.

Another challenge is that notations for service descriptions depend on perspective. Service descriptions on a business operational perspective and a technology perspective differ in concepts and semantics. A system to categorize different service description notations and realization languages would support a common understanding. Service engineering would benefit from such a classification, since it comprises different stakeholders and different phases such as innovation, strategic planning, service design, service implementation, and market launch [11]. It would support the work to derive service description from service provider's business models, and to implement them with web service related technology.

This paper presents (1) service properties and their relationship to specify a service description from a service provider's viewpoint, and (2) an adaptation for the Zachman Framework [31] in order to categorize service description notations for various perspectives.

The remainder of this paper is structured as follows: section two introduces service ecosystems. The Zachman Framework is summarized in section three. The service properties and their relationships are explained in section four, and section five presents the service description adaptation for the Zachman Framework. Section six concludes this paper. Related work is presented in the relative sections.

2 Service Ecosystems

Web services [34] for heterogeneous application integration and communication between companies gained popularity during the last years [27]. Recently, companies, such as Amazon.com, acknowledged web services beyond integration as a means to create value for customers. In consequence, the service ecosystem concept gained momentum. Since it is a fairly new field of research, various names exist for service ecosystems, which include service systems [32] and internet of services [25]. The vision of service ecosystems is an evolution of service orientation and takes services from merely integration purposes to the next level by making them available as tradable products on service delivery platforms [4].

Service ecosystems are market places for trading services in the business sense and involve actors from different legal bodies. Service trade involves the following steps: service discovery, service selection, service contracting, service consumption, monitoring, and profiling. During discovery and selection, service providers advertise their services toward potential consumers, whereas service consumers specify their service preferences toward providers. During service contracting, providers and consumers negotiate and finally agree on service levels (SLA) which are monitored (for billing & payment) throughout service consumption. In the event service levels are not met, compensations must be triggered. During service profiling, valuable information on services' performance is stored, which is gathered during consumption and monitoring.

On the technical side, service ecosystems refer to a logical web service collection [4], with more than one service provider. On the business side, service

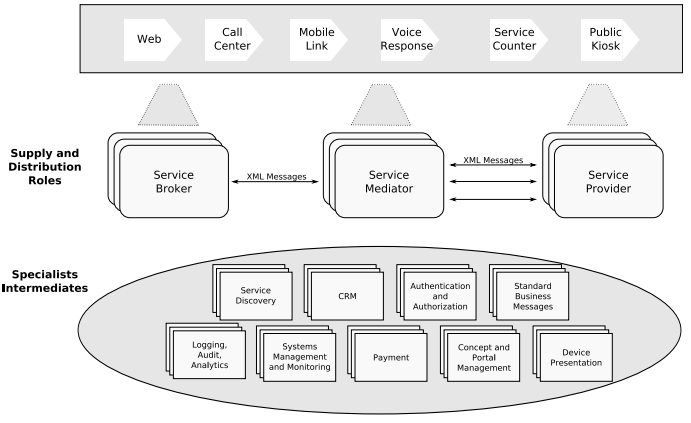


Fig. 1. Top-level Architecture of a Service Ecosystem [4]

ecosystems bring together shared information, people, and technology [32]. They comprise service innovation, service design, service engineering, service marketing, and service provisioning [11]. These systems provide core services such as payment and monitoring, domain-specific services such as eco-value calculations [9], and complex services such as travel services. These service are leveraged by others to implement end-to-end business processes [28], which cross companies' borders, to create value for end-customers. Business models such as Business Webs [33] foster this idea of *coopetition*.

Barros and Dumas [4] (cf. figure 1) identify next to service consumers three different roles for actors in service ecosystems. *Service providers*, who provide services in the first place. *Service brokers* offer services from different providers. Their business model is to bring providers and consumers together, or enhance services with delivery functions for convenient service provisioning. *Service mediators* generate value by customizing provider's standard services toward consumer's needs. All the same, service ecosystem actors may play more than one role.

Services that are traded in service ecosystems are so-called *e-services*. Baida et al. [2] offer a terminology comprising the terms service, e-service, and web service. Services refer "... to business activities which often result in intangible outcomes or benefits ..." [2]. E-services, in contrast, refer to service provisioning by means of electronic network protocols, such as the Internet. E-services are technically implemented with web services. For the rest of this paper, the term service refers to the e-service definition.

In conclusion, service ecosystems aim at (1) trading services over the internet, (2) composing complex services from existing ones, and (3) supporting service provisioning with IT [9].

3 Zachman Framework

The Zachman Framework [35] provides a taxonomy to relate real world concepts to Enterprise Architecture [31]. Zachman describes Enterprise Architecture as

means to flexibly react to business changes and to manage the varied resources of an enterprise. The Zachman Framework embodies vital artifacts to describe, create, operate, and change an object. The term "Object" is used consciously, since it may relate to practically anything, e.g., an enterprise, a project, a solution, and in this case a service.

The Zachman Framework distinguish between six perspectives and six descriptions which are orthogonal to each other. The first six columns (internal view) in figure 2 depicts the framework's matrix. Each column of the matrix offers a basic model for the description in question from a certain perspective. It is important to note that the framework does not specify the order of descriptions. Each intersection is a placeholder for a basic notation which satisfies a columns' basic model.

3.1 Perspectives

The six different perspectives are organized into corresponding layers [31]. It is important to note that the various perspectives are different with respect to nature, content, and semantics and not only in their detail level [35].

The *scope layer* represents the planner's perspective. The purpose of this layer is to identify "... the size, shape, spatial relationships, and final purpose of the final structure." [31] and thus, the scope. On this basis a planner decides whether to invest in the architecture.

The *business layer* symbolizes the owner's perspective. Architects describe the requirements from the owner's perspective, whereas the intention is to "... enable the owner to agree or disagree with the ..." [35] description.

The *system layer* corresponds to the designer's perspective. The purpose of this layer is to transform the enterprise model's artifacts into detailed specifications. The owner can use these specifications to negotiate with builders to implement the system.

The *technology layer* represents the builder's perspective. The rationale of this layer is that the detailed specifications must be adapted into builder's plans to take into account the "... constraints, of tools, technology, and materials." [31].

The *component layer* symbolizes the perspective of a sub-contractor. Builder's plans are translated into shop plans. Shop plans "... specify details of parts or subsections ..." [31] of builder's plans.

The *operations layer* represents the system itself.

3.2 Descriptions

The six descriptions depict an enterprise from different angles. Though, each of them is unique and addresses a different purpose, they relate to each other [35]. Descriptions are the answers to the basic questions: What (Data Description), How (Process Description), Where (Location Description), Who (People Description), When (Time Description), and Why (Motivation Description). It is important to note, that for each description exists a set of terms (description model) which are valid for all perspectives. Nonetheless, these terms differ essentially in semantics for each perspective.

The *data description*'s model consists of entities and relationships between entities. The basic intention is to identify enterprises' inventory.

The *process description*'s model embodies processes and arguments (input and output to processes). The purpose is to make out enterprises' processes and business functions.

The *location description*'s model uses the concepts of locations and connections in order to discover enterprises' network.

The *people description*'s model is that of roles and work. The description's intention is the "... allocation of work and the structure of authority and responsibility." [31].

The *time description*'s model embodies event and cycle. The description's intention is to "... produce a schedule of events and states that maximizes the utilization of available resources while at the same time satisfying the external commitment." [31].

The *motivation description*'s model uses the concepts of ends and means. The motivation description's intention is to describe the motive of an enterprise, where ends equal objectives and means equal strategies.

4 Service Properties

This section introduces service properties to describe services. Initially, Scheithauer and Winkler [26] introduced these properties. They collected properties from different sources: (1) PAS 1018:2002 [14], (2) IEEE 830:1998 [29], (3) IEEE 1061:1998 [30], (4) ISO 9126 [8], (5) O'Sullivan's service properties [20], (6) quality attributes [3]. The actual set presents some adapted properties as well as a different and a more coherent property grouping. The taxonomy from Avizienis et al. [1] was added to the analysis for quality of service. Additionally, a hierarchy is introduced which depicts the parent property in combination with cardinalities. Nevertheless, the presented properties are not intended to be complete. They rather show the current state of the work. Once service ecosystems' requirements toward describing services are discovered, an exhaustive analysis is possible. Another limitation is that the presented properties lack appropriate metrics. This results from the fact that depending on the perspective, properties are interpreted differently, e.g., a capability is differently expressed on the scope layer than on the technical layer. This challenge is beyond the scope of this paper.

These properties are valid for all perspectives of the Zachman Framework (cf. section 5). It is important to note that these properties do not intend to describe services' behavior, implementation, nor how to technically integrate a service into various software environments. They rather serve to propose a service on a market place toward potential customers in terms of functionality, financial, legal, marketing, and quality aspects. The following subsections will shortly introduce each property.

Table 1. Functional Service Properties

Property Name	Parent Property	Cardinality
Capability	Service Description	1...*
Classification	Service Description	0...*

4.1 Functionality

Functionality provides the service consumer with an understanding of what the service is actually providing and thus, what the consumer can expect from the service. Properties include capabilities and service classifications (cf. table 1).

A *Capability* is the major function that services provide. A capability allows a service consumer to access services' functionality. Often, services' functionality is divided into several capabilities. This allows service consumers to access particular subsets of services' functionality. Additionally, a service's outcome might be different, depending on capabilities to perform in what order. E.g., a flight booking service offers different capabilities, such as to browse different flights, to plan a flight route, to book a flight, and to pay for it. In some cases just some of these capabilities are necessary for service consumers to achieve their goals. However, to book a flight, one of more specific capabilities must be invoked in a predefined way. A service has one or more capabilities. This property is represented by a formal naming system. This would include a capability's name, involved parties, data which is processed by the service, and the outcome, which address also pre- and postconditions.

A *classification* allows to apply the service into one or more classification systems. A classification is a system of interrelated terms which generally form a hierarchical structure. The terms allow to specify the kind of service, an unique identifier, and a reference to a classification standard, such as eCl@ss and UNSPSC [7]. While the classification property is optional, it may be the case that a service is classified according to multiple classification standards. For that reason it is necessary to model service classification as a tuple of a reference to a classification standard and a unique identifier.

4.2 Financial

This section comprises monetary related properties (cf. table 2).

The *price* property represents an economical numerical value for services. PAS 1018:2002 [15] and O'Sullivan [20] list this property. PAS 1018:2002 depicts two price properties. The first price property describes service providers' price conception. The second price property specifies the service consumers' price idea. O'Sullivan, however, offers a more holistic approach. His work includes four different types of price. It is possible to relate all price types to entities such as *time*, *area*, etc. This allows to specify different prices for different time or areas of service usage. Additionally, tax information can be included as well. Four price types are explained briefly [20]. An *absolute price* specifies a specific amount of money and a currency. E.g., booking a flight costs EUR 10. A *proportional*

Table 2. Financial Service Properties

Property Name	Parent Property	Cardinality
Absolute Price	Capability	0...*
Ranged Price	Capability	0...*
Proportional Price	Capability	0...*
Dynamic Price	Capability	0...*
Discount	Price	0...*
Early Payment	Discount	0...*
Payment Instrument	Discount	0...*
Coupon	Discount	0...*
Volume	Discount	0...*
Location	Discount	0...*
Age Group	Discount	0...*
Student	Discount	0...*
Membership	Discount	0...*
Shareholder	Discount	0...*
Payment	Service Description	1...*
Payment Option	Payment	1...1
Payment Schedule	Payment	0...*
Card Instrument	Payment	1...*
Cheque Instrument	Payment	1...*
Cash Instrument	Payment	1...*
Voucher Instrument	Payment	1...*

price depicts a percentage with respect to a given value. E.g., a life insurance monthly rate is 1% of one's yearly income. A *ranged price* allows to specify a price range with a minimum and maximum *absolute price* or *proportional price*. Service providers may use this price type in case it is impossible to set an absolute price. Fixing the final price is part of the negotiation phase between service providers and service consumers. E.g., a rental car's price per day ranges from EUR 50 to EUR 70. The final price depends on the final car configuration. A *dynamic price* covers auctions, where the price matching is based on natural supply and demand. E.g., a service provider offers train tickets and potential service consumers bet an amount of money they perceive as their value. The metric for currencies is the ISO 4217:2001. The price amount is represented by a numerical data type.

The *payment* property specifies feasible options to fulfill service consumer's payment liability. PAS 1018:2002 [15] and O'Sullivan [20] list this property. Where PAS 1018:2002 depicts only a placeholder for payment, O'Sullivan offers a more thorough approach. However, they do not contradict each other. According to O'Sullivan [20], payment is complementary to the *price* property. He subdivided this property into four models: payment options, payment schedules, payment instruments, and payment instrument types. A *payment option* constitutes, whether a particular payment option is the preferred one, whether there is a charge connected to the payment option, where a payment option is available, specific conditions for a payment option, and currencies. A *payment schedule* depicts when a

payment is due. This property has two dimensions. Firstly, it is possible to specify a percentage of the whole price with respect to services' provisioning moment (before, during, and after). Secondly, percentages together with concrete dates can be specified. Four *payment instrument types* are available: *card based instruments*, *cheques*, *cash*, and *vouchers*. A service has one or more payment properties. Each payment property has exactly one option, none or more schedules, and at least one instrument. Payment is a mandatory property. Dates are represented with ISO 8601, currencies with ISO 4217, and regions with ISO 3166.

The *discount* property specifies possible price reductions. Only O'Sullivan [20] lists this property. In general, *discount* properties can be offered within a specified time segment (temporal), for a specific location (locative), or a given condition. The discount property is differentiated between payment related discounts and payee related discounts. *Payment related discounts* group types of discounts that refer to how payment is done. This includes *early payment*, *type of payment instrument*, *coupons*, *location of payment*, and *volume invocation*. *Payee related discounts* relates to the service consumer, who pays for a service. This includes *age group*, *student*, *membership*, and *shareholder*. A service has no or more discounts for a *price*. Dates are represented with ISO 8601, and regions with ISO 3166.

4.3 Legal

The legal category embodies properties which state terms of use. The properties are rights, obligations, and penalties (cf. table 3).

Rights state what service consumer are allowed or expected to do with the service. For service ecosystems, re-selling, and re-bundling with other services needs to be considered. Rights are represented with semantically defined terms.

Obligations determine and settle the commitment for a service provider and a service consumer. This includes what a service provider must deliver. Obligations are represented with semantically defined terms.

The *penalty* property dictates compensation in case an obligation was not met by one party. Each obligation property relates to one penalty property to cover the effects. Penalties are represented with semantically defined terms.

Table 3. Legal Service Properties

Property Name	Parent Property	Cardinality
Right	Service Description	0...*
Obligation	Service Description	0...*
Penalty	Obligation	1...1

4.4 Marketing

The marketing category allows to promote the service toward potential customers. Properties in this category should both attract customers and establish

Table 4. Marketing Service Properties

Property Name	Parent Property	Cardinality
Certification	Service Description	0...*
Expert Test Rating	Service Description	0...*
Benefit	Service Description	0...*

a trusted relationship. A certification would provide a rather neutral view on a service provided by a third party. On the other hand, expert test ratings provide a subjective view on the service from an expert perspective. Service benefits are the gained outcome of the service with respect to the potential service consumer. These properties are summarized in table 4.

The *certification* property represents a declaration issued by trusted institutes or by the service platform itself. This property tells whether a service is certified by a known and trusted party. This party issues a certificate in case one or more requirements regarding services are met. An analogous concept is the certification for secure websites. The certificate is represented with a formal system or a common standard, such as the *X.509*.

Expert test rating represents a rating from autonomous parties which are experts in the service domain. Potential service consumers might consult the expert test rating to decide whether to use the service or not. The expert test rating is determined by thorough tests, where domain-specific criteria are applied to services and then, depending on the performance, are rated. This property may be represented via a scale of values ranging from a minimum to a maximum value (e.g., scale from 1 to 10 as described before).

The *benefit* of a service is the gained outcome of the service for the service user. This information is needed for a potential service consumer to determine whether this particular service has the potential to suit its needs.

4.5 Quality of Service

As aforementioned, service provisioning in service ecosystems is conducted over the internet, technical properties of the network and the service itself can be of importance for service discovery and selection. Properties include: (1) performance and (2) dependability (cf. table 5).

Performance represents a service's responsiveness with respect to events and time [3]. Performance is expressed with *latency* and *throughput*. Latency describes a fixed time interval in which an expected event's response must be fired. Throughput specifies how many responses to a given event during a time interval will be completed.

Avizienis et al. define [1] dependability as "... the ability to deliver service that can justifiably be trusted." For this analysis, *availability* and *reliability* are regarded. A service's availability represents the systems readiness to be triggered [3]. Reliability, on the other hand, express the service's capability to keep performing over time [3]. Both, availability and reliability are expressed with a percentage.

Table 5. Quality Service Properties

Property Name	Parent Property	Cardinality
Latency	Service Description	0...1
Throughput	Service Description	0...1
Availability	Service Description	0...1
Reliability	Service Description	0...1

5 Service Description for the Zachman Framework

This section presents a seventh description for the Zachman Framework to create a classification for service description notations and realization languages. The general logic for *perspectives* (cf. section 3) is used as the basis.

The existing six descriptions in the Zachman Framework address the solution itself, e.g., an enterprise architecture or a service. These descriptions can be considered as an *internal view* on the solution. Considering service ecosystems as market places to trade services, an internal view is not appropriate for two reasons: (1) service customers might not be interested in how services work or are implemented and (2) service providers do not want to expose this information to customers and competitors. Thus, a description must become available which depicts the *external view* of the solution, which promotes the service toward potential customers. This *external view* has different semantics depending on the different perspectives during service engineering.

Hence, the Zachman Framework is used to align the service description notations and realization languages with the six different perspectives. This allows to find appropriate service description notations and realization languages for each perspective and to identify the need for further notations. Furthermore, this classification supports the arrangement of the perspectives and their appropriate notations to foster business-IT alignment.

One limitation to this approach is that this adaptation works for service ecosystems. In other environments than service ecosystems, different requirements might apply, and therefore other adaptations than the one presented become necessary.

It is important to note that the service properties introduced in section 4, are valid for all perspectives of the Zachman Framework.

The *service description*'s model is that of properties and values. The service description intention is to describe a service's proposition [6] a company offers toward its customers. *Properties* refer to service elements which describe a certain aspect of services. *Values* refer to the characteristic of service elements. This model is valid for each perspective, though the semantic differs for each of them. Figure 2 depicts the resulting framework.

On the *scope layer* properties have a strategic semantic and take into account the service final purpose and context by listing the important properties. For example, the price property refers to a price strategy, and price values refer to a specific strategy, such as Porter's generic price strategies. Suitable models for this perspective are the business model ontology [19,25] and the service bundle [22].

	Internal View						External View
	Data	**Process**	**Location**	**People**	**Time**	**Motivation**	**Service**
Scope Layer	List of important Entities	List of important Processes	List of important Locations	List of important Organizations	List of important Events	List of important Goals and Strategies	List of important Properties
Business Layer	Semantic Model	Business Process Model	Business Logistic Model	Workflow Model	Master Schedule	Business Plan	Value Proposition
System Layer	Logical Data Model	Application Architecture	Distributed System Architecture	Human Interface Architecture	Processing Structure	Business Rule Model	Service Model
Technology Layer	Physical Data Model	System Design	Technology Architecture	Presentation Architecture	Control Structure	Rule Design	Service Profile
Component Layer	Data Definition	Program	Network Architecture	Security Architecture	Timing Definition	Rule Specification	Service Component
Operations Layer	DATA	FUNCTION	NETWORK	ORGA-NIZATION	SCHEDULE	STRATEGY	SERVICE

Fig. 2. Service Adaptation for the Zachman Framework

Properties and values on the *business layer* describe the owner's requirements with respect to the services. The result is a value proposition toward potential customers. For example, the price property refers to a price model, and the price value to a specific price model, such as Proportional Price [20].

On the *system layer* designers create a complete service model, which is technology-independent and formal. On this formal basis, builders can implement a platform-dependent service description. For example, the price property refers to the obligation a potential service consumer must pay in order to consume the service and price value specifies the amount of money. Suitable model notations for this perspective are the UML Profile and Metamodel for Services (UPMS) [17], the UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms (UPMQoS) [16], as well as the Service Component Architecture (SCA) [18].

Properties and values on the *technology layer* are adapted to a concrete technology. Appropriate technologies for web services include the Web Service Modeling Ontology (WSMO) [24] and Web Ontology for Services (OWL-S) [13].

On the *component layer* the service description is divided into sections. This parting allows to realize different parts independently. For example functionality properties and values are presented with a WSDL file [23], and quality of service properties and values with the web service level agreement (WSLA) [10].

The *operation layer* represents the implemented service description itself.

6 Conclusion and Future Work

First, service properties were presented to describe services' external view. Second, an adaptation for the Zachman Framework was motivated and introduced. Lastly, notations were selected for each perspective. Service providers are now in the position to categorize different service description notations with respect to the various perspectives which are involved during service engineering. Nonetheless, nothing is said about method. A perspective's service description can be either modeled *before* the other descriptions as a requirement, or *after* the other descriptions. The service properties offer attributes to describe services in a common way and hence narrow the gap between business model's value propositions and service description implementations. It serves as a domain-specific taxonomy for describing services for service ecosystems.

Future work includes the evaluation of the service properties. Additionally, requirements for service descriptions must be derived from service ecosystems to improve and complete the service property set. Furthermore, the service properties will be codified in a formal language, in order to share this knowledge between service providers and service consumers. The Zachman Framework adaptation is already in place with the Inter-enterprise Service Engineering (ISE) methodology [11]. Tool support is under development. Finally, a routine needs to be developed to model and to derive service properties from business model's value propositions [25].

Acknowledgments

This project was funded by means of the German Federal Ministry of Economy and Technology under the promotional reference "01MQ07012". The responsibility for the content of this publication lies with the authors. The presented service properties in this paper are also a result from previous work [26] which was done in collaboration with Matthias Winkler from SAP Research.

References

1. Avizienis, A., Laprie, J.-C., Randell, B.: Dependability and its threats - A taxonomy. In: Jacquart, R. (ed.) IFIP Congress Topical Sessions, pp. 91–120. Kluwer, Dordrecht (2004)
2. Baida, Z., Gordijn, J., Omelayenko, B.: A shared Service Terminology for Online Service Provisioning. In: Janssen, M., Sol, H.G., Wagenaar, R.W. (eds.) ICEC. ACM International Conference Proceeding Series, vol. 60, pp. 1–10. ACM, New York (2004)
3. Barbacci, M., Klein, M.H., Longstaff, T.A., Weinstock, C.B.: Quality attributes. Technical Report ESC-TR-95-021, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213 (December 1995)
4. Barros, A.P., Dumas, M.: The Rise of Web Service Ecosystems. IT Professional 8(5), 31–37 (2006)

5. Booms, B., Bitner, M.: Marketing strategies and organization structures for service firms. In: Marketing of Services, American Marketing Association, Chicago, IL, pp. 47–51 (1981)
6. Gordijn, J., Petit, M., Wieringa, R.: Understanding business strategies of networked value constellations using goal- and value modeling. In: RE, pp. 126–135. IEEE Computer Society, Los Alamitos (2006)
7. Hepp, M., Leukel, J., Schmitz, V.: A quantitative analysis of product categorization standards: content, coverage, and maintenance of ecl@ss, UNSPSC, eOTD, and the rosettanet technical dictionary. Knowl. Inf. Syst. 13(1), 77–114 (2007)
8. International Organization for Standardization (ISO). ISO/IEC 9126: Software engineering - Product quality (2001)
9. Janiesch, C., Ruggaber, R., Sure, Y.: Eine Infrastruktur fuer das Internet der Dienste. HMD - Praxis der Wirtschaftsinformatik 45(261), 71–79 (2008)
10. Keller, A., Ludwig, H.: The WSLA framework: Specifying and monitoring service level agreements for web services. J. Network Syst. Manage 11(1) (2003)
11. Kett, H., Voigt, K., Scheithauer, G., Cardoso, J.: Service Engineering for Business Service Ecosystems. In: Proceedings of the XVIII. International RESER Conference, Stuttgart, Germany (September 2008)
12. Kuropka, D., Troeger, P., Staab, S., Weske, M. (eds.): Semantic Service Provisioning. Springer, Heidelberg (2008)
13. Martin, D., Paolucci, M., McIlraith, S.A., Burstein, M., McDermott, D., McGuinness, D.L., Parsia, B., Payne, T.R., Sabou, M., Solanki, M., Srinivasan, N., Sycara, K.P.: Bringing semantics to web services: The OWL-S approach. In: Cardoso, J., Sheth, A.P. (eds.) SWSWPC 2004. LNCS, vol. 3387, pp. 26–42. Springer, Heidelberg (2005)
14. Mörschel, I., Behrens, H., Fähnrich, K.-P., Elze, R.: Standardisation in the Service Sector for Global Markets. In: Advances in Services Innovations, Engineering, pp. 257–277. Springer, Heidelberg (2007)
15. Mörschel, I.C., Höck, H.: Grundstruktur für die Beschreibung von Dienstleistungen in der Ausschreibungsphase. Beuth Verlag GmbH, Ref. Nr. PAS1018:2002-12 (2001)
16. Object Management Group (OMG). Specification: UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms, v 1.1 (May 2005), http://www.omg.org/technology/documents/formal/QoS_FT.htm
17. Object Management Group (OMG). Specification: UML Profile and Metamodel for Services RFP UPMS Services Metamodel (September 2006), http://www.omg.org/cgi-bin/doc?soa/2006-09-09
18. Organization for the Advancement of Structured Information Standards (OASIS). Specification: Service Component Architecture (SCA), http://www.oasis-opencsa.org/sca
19. Osterwalder, A.: The Business Model Ontology: A Proposition in a Design Science Approach. PhD thesis, Universite de Lausanne Ecole des Hautes Etudes Commerciales (2004)
20. O'Sullivan, J.: Towards a Precise Understanding of Service Properties. PhD thesis, Queensland University of Technology (2006)
21. Peneder, M., Kaniovski, S., Dachs, B.: What follows tertiarisation? structural change and the role of knowledge-based services. The Service Industries Journal 23(2(146)), 47–66 (2003)
22. Pijpers, V., Gordijn, J.: Bridging Business Value Models and Process Models in Aviation Value Webs via Possession Rights. In: HICSS, p. 175. IEEE Computer Society, Los Alamitos (2007)

23. Chinnici, A.R.S.W.R., Moreau, J.-J.: Specification: Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. W3C Recommendation, 6 (2007)
24. Roman, D., Keller, U., Lausen, H., de Bruijn, J., Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, C., Fensel, D.: Web Service Modeling Ontology. Applied Ontology 1(1), 77–106 (2005)
25. Scheithauer, G.: Process-oriented Requirement Modeling for the Internet of Services. In: Ruggaber, R. (ed.) Proceedings of the 1st Internet of Services Doctoral Symposium 2008 (I-ESA), Berlin, Germany, vol. 374, March 25 (2008)
26. Scheithauer, G., Winkler, M.: A Service Description Framework for Service Ecosystems. Bamberger Beiträge zur Wirtschaftsinformatik 78, Bamberg University (October 2008) ISSN 0937-3349
27. Scheithauer, G., Wirtz, G.: Applying Business Process Management Systems – a Case Study. In: The 2008 International Conference on Software Engineering and Knowledge Engineering (SEKE 2008), Redwood City, California, USA, pp. 12–15 (2008)
28. Scheithauer, G., Wirtz, G., Toklu, C.: Bridging the Semantic Gap between Process Documentation and Process Execution. In: The 2008 International Conference on Software Engineering and Knowledge Engineering (SEKE 2008), Redwood City, California, USA (2008)
29. Software Engineering Standards Committee of the IEEE Computer Society USA. IEEE Guide for Software Requirements Specifications, 830-1998 (1998)
30. Software Engineering Standards Committee of the IEEE Computer Society USA. IEEE Standard for a Software Quality Metrics Methodology, 1061-1998 (1998)
31. Sowa, J.F., Zachman, J.A.: Extending and Formalizing the Framework for Information Systems Architecture. IBM Systems Journal 31(3), 590–616 (1992)
32. Spohrer, J., Maglio, P.P., Bailey, J., Gruhl, D.: Steps toward a science of service systems. IEEE Computer 40(1), 71–77 (2007)
33. Tapscott, D., Ticoll, D., Lowy, A.: Digital capital: harnessing the power of business Webs. Harvard Business School Press (May 2000)
34. W3C Working Group. Web services glossary (February 2004), http://www.w3.org/TR/ws-gloss/
35. Zachman, J.A.: A Framework for Information Systems Architecture. IBM Systems Journal 26(3), 276–292 (1987)

Service Selection in Business Service Ecosystem

Sujoy Basu, Sven Graupner, Kivanc Ozonat, Sharad Singhal, and Donald Young

Hewlett-Packard Laboratories, 1501 Page Mill Road
Palo Alto, CA 94304, USA
{sujoy.basu,sven.graupner,kivanc.ozonat,sharad.singhal}@hp.com

Abstract. A world-wide community of service providers has a presence on the web, and people seeking services typically go to the web as an initial place to search for them. Service selection is comprised of two steps: finding service candidates using search engines and selecting those which meet desired service properties best. Within the context of Web Services, the service selection problem has been solved through common description frameworks that make use of ontologies and service registries. However, the majority of service providers on the web does not use such frameworks and rather make service descriptions available on their web sites that provide human targeted content.

This paper addresses the service selection problem under the assumption that a common service description framework *does not* exist, and services have to be selected using the more unstructured information available on the web.

The approach described in this paper has the following steps. Search engines are employed to find service candidates from dense requirement formulations extracted from user input. Text classification techniques are used to identify services and service properties from web content retrieved from search links. Service candidates are then ranked based on how well they support desired properties. Initial experiments have been conducted to validate the approach.

Keywords: Service, selection, unstructured data, service discovery, service matchmaking, text analysis, service ontology, service description framework.

1 Introduction

The service selection problem is how to *find and select services* offered by service providers that best meet the requirements described by a service seeker. On one side, the service seeker provides a description of requirements or desired service properties. On the other side, service providers describe their capabilities. In general, this is a two-step process. In the first step, a set of service provider candidates, which generally meet requirements (not necessarily all requirements), are identified. Requirements are described as a set of desired service properties. In the second step, the "best" service provider is selected from the candidate set based on a "best match" between the properties exposed by the service provider and those requested by the service seeker. Typically, in order to solve the service selection problem, a common base must exist or must be established between the descriptions of desired service properties on one side and descriptions of service capabilities on the other. This common base can be strong and formal, as it has traditionally been defined by *service*

G. Feuerlicht and W. Lamersdorf (Eds.): ICSOC 2008, LNCS 5472, pp. 256–269, 2009.

description frameworks with pre-defined syntactic and semantic constructs for expressing service properties and query and matchmaking capabilities built on top of it. A service registry is an example of a system which defines a service description framework.

While the service selection problem has been explored and has largely been solved under the assumption that a formal service description framework exists, this is not true of the majority of service offerings, which are described in natural language in marketing collateral, web content or advertisements. The same applies at the service seeker's side: most documents that describe what a person is looking for are expressed in (largely) unstructured formats. And this is particularly true for the web as the most significant medium through which service offerings are promoted and advertised as well as sought and found. Consequently, the service selection problem must be considered for an unstructured environment such as the web.

We see a need to enable service technology to better deal with unstructured data. It will be a central challenge for the comings years. In this paper, we present an approach to the service selection problem using unstructured data for service properties. We address the problem under the assumption that no common description framework exists. In the proposed approach, first, search engine is employed to find service candidates from dense requirement formulations extracted from user input. Next, text classification techniques are used to identify services and service properties from web content retrieved from returned search links. Service candidates are then ranked based on how well they supported desired properties. We have evaluated the approach through conducting experiments on a sampling of real requirements documents of an internal marketing department of a large corporation. The experiments show encouraging results.

The paper is structured as follows. First, we discuss related work in Section 2. Then we discuss assumptions underlying the work in Section 3. Next, we present the problem statement in Section 4. Section 5 gives an overview of the approach of the proposed solution. Details about the techniques and experiments are presented in Section 6. Finally, we discuss future work and conclude the paper in Section 7.

2 Related Work

Service technology from the Web 1.0 era mainly relied on established service description frameworks in order to find and select services. Prominent representatives are service registries, which are a fundamental building block of Service-Oriented Architecture (SOA) [1] today. An example is the Universal Description, Discovery and Integration registry (UDDI) [2]. Service registries have not only been defined for technical (web-) services, but also for businesses services (e.g. OASIS ebXML Registry [3]). Registries define syntax and semantics in which service descriptions must be expressed. A common pattern is the use of attribute-value pair sets. Query techniques in registries rely on either a priori known or discoverable attribute sets for finding and selecting services. Query languages are based on Boolean expressions or first-order logic upon property sets. They lead consistently to defined, repeatable search results in a given search space. More advanced techniques use extendable service vocabularies and ontologies for expressing and matching service properties. Service technology of this kind has been targeted towards machines.

Approaches for discovering Web services, i.e., those that are described using WSDL [19] interfaces, are extensively investigated in the literature (e.g., [20], [21], [22]). There exist also Web services repositories and portals where such services are indexed and can be explored and discovered (e.g., XMethods [23] and Seekda [24]). in the focus of this paper is rather on general *services over the Web*, most of which are described in natural language. Indeed, more recent service technology is, in contrast, more targeted towards both people as service seekers and providers. The common base here is a common natural language with all its ambiguity making it hard for machines to identify good service selections.

The Web plays a central role today for bringing service seekers and service providers together, as individuals or as businesses. An indicator for this is the significant revenue search engines harvest from searches on the web. And yet, support for service selection in search engines and on the web in general remains rather limited. More advanced techniques are needed which can deal with the ambiguity of the web.

Ideally, the web should "understand" human natural language, and there are efforts towards this [4, 5]. Similarly, efforts are underway in the Semantic Web [6] to structure web content. However, neither has made the necessary progress. Rather, text analysis and categorization techniques appear more practical and promising and have widely been explored on general web content [7, 8, 9], but not specifically for the service selection problem. This paper specifically considers the service selection problem and outlines an approach that uses text analysis techniques for finding and selecting services offered over the web.

3 Assumptions

The paper addresses the service selection problem from the perspective of a seeker of services in a corporate or private environment. It is assumed that someone (a person) in the service seeker's role informally knows about a task to be given to a service provider and that a textual description of desired service properties exists. One form of expressing these properties is in form of documents such as a statement-of-work (SOW), a request-for-proposal (RFP) or a request-for-quote (RFQ) document that corporations use to procure external services. We use a sample set of these documents for identifying desired service properties.

It is furthermore assumed that service providers have presences on the web (web sites) where they informally describe and advertise their capabilities and that those web pages can be found through search engines. While it is assumed that informal descriptions about service providers can be obtained from the web, it is *not* assumed that the actual business interaction also is mediated over the web. For example, a legal counseling agency may be found on the web, but actual counseling then occurs in person. We explore service providers' web content and classify its properties.

Another assumption is that a common language (English) is used to describe sought and offered service properties; and that the same words or phrases are used for same concepts. WordNet [10] provides a rich set of English language words, terms, phrases and defining semantics.

4 Problem Statement

Based on these assumptions, the paper addresses the following problems:

1. Can sought service properties (requirements) be gathered informally from a business person in a service seeker role and represented in a condensed descriptive vector of meaningful terms?
2. Can these terms then be used in search engines to find service provider candidates? This includes that service providers must be distinguished from other content returned from search.
3. Can service properties be extracted and classified from service providers' web content (their web sites)?
4. Can properties from service seeker's requirements and service provider's capabilities be correlated such that providers can be ranked based on how well they support requirement properties?

The first two problems relate to "how to find service candidates"; the last two address the matchmaking aspect of the service selection problem.

5 Approach

People seeking services typically go to the web and search for them and then select what they like. This process is inherently manual since most data based on which those decisions are made exists as unstructured data, e.g. content of web pages. In this paper, an approach has been developed which allows automating the process.

A search engine is employed to find service candidates. It is fed with key words which are extracted from informal user input using a text classification technique. The result page returned from the search engine contains links to potential service providers. Those links are followed programmatically to fetch content which may or may not represent a service provider. A machine learning technique is employed to automatically make a judgment whether content represents a service provider or not. Furthermore, if content was classified as representing a service provider, more pages are fetched from this site to accumulate enough material to again employ a text analysis technique to determine how well desired service properties are supported. The produced result is factored into a final score of all identified service providers to rank order them.

The approach consists of four steps, each addressing one of the above problems:

The first step aims at condensing the information business people use informally when communicating service requirements. The goal is to identify the essential terms from documents which describe required service properties. Forms and questionnaires are familiar to a business audience and a good way to produce dense information for procuring external services. An example illustrates this approach. For a marketing campaign at a larger customer event, a service provider may be sought which can *"print quantity 2,000 colored 8.5x11 z-fold brochures, 100 lb. gloss within 10 days with maximum budget of $1,000"*. This string represents a condensed form of a statement of work document and the essence of desired service properties. Latent Semantic Indexing (LSI) [11] is used to extract these terms from a representative set of documents. This

step results in a descriptive vector of meaningful words representing the essence of required service properties.

The second step is to use these meaningful words in search engines and to obtain a pool of links to potential service candidates. Since links returned from search may refer to any content, which may or may not be service providers, links must be followed and content obtained from links in order to determine whether or not content represents a service provider. If content could be successfully probed and classified as representing a service provider, the service provider is added the pool of potential service provider candidates. For example, when the string above is typed into the Google search engine, it returns 11 sponsored links of print service providers (in the US) and a number of random entries, which are more or less related to printing brochures, but may or may not be service providers. Typically, in Google, it is sufficient to consider content returned with the first page. Other search engines such as Yahoo! and Microsoft may return different content.

Further sources of information about service providers can be involved such as established service catalogs such as Hoovers [12], Dun and Bradstreet [13] or ThomasNet [14] in order to obtain a larger candidate pool of service providers. These service catalogs have been collecting structured information about businesses in the US and worldwide and make this information available over the web.

However, the answer from search engines or service catalogs can only be used as a starting point to further explore whether or not a returned link represents a service provider. The second problem hence addresses whether or not a site behind a link can be identified as a service provider. The approach here is to look for FORM pages encouraging users to engage with the service. This step results in a pool of potential service provider candidates.

Furthermore, in preparation of comparison, service properties must be identified for candidates from their web content. The approach here relies on meta-tags and content of online service engagement forms. Thus, this step also provides a set of service properties identified for each service.

The service candidates are ready for comparison after their service properties have been extracted. Furthermore, they must be correlated with service properties from condensed requirements derived in the first step. Since LSI has been used in the first step, we repeat the process with the service provider pages to generate their condensed set of properties as a vector of terms. Then we use cosine similarity as the metric to rank the list of service provider candidates that support the desired service properties best.

While this process does not guarantee that service providers found as a result will support the requirements and that the top-ranked service candidate is indeed the best to engage, it mirrors the process a person would follow when asked to find a product or a service on the web. The advantage here is that we can automate the manual steps to potentially explore a much larger set of service provider candidates than would be feasible manually. In the next section, we describe a number of initial experiments that have been conducted to highlight the potential of this approach.

6 Techniques and Experiments

This section describes experiments that have been conducted for the four steps. We consider these experiments as an initial set of experiments and are currently scaling

up our experiments to larger data sets. We do not claim that these experiments represent the best choices of techniques that could have been made, nor do we claim to evaluate and compare different techniques in this paper. Our goal is to demonstrate a feasible implementation of the four steps of the service selection problem in an unstructured web environment. Others researchers such as Forman [15] have done extensive empirical study of feature selection for text classification.

6.1 Extracting Significant Words for Service Requirements

Input. We use 24 PDF documents from an internal marketing department, which are primarily request for quotes (RFQ) for various printing projects undertaken by a large enterprise typically through an entity known as the print concierge. The RFQs are normally sent to approved vendors.

Objective. We seek an automated method, which is not based on domain-specific knowledge, which can identify the list of terms representing the essence of required service properties and can handle synonymy and redundancy inherent in natural language documents. We do not expect the number of RFQ documents to grow as fast as documents on the Web. We are currently using Latent Semantic Indexing (LSI) to identify the list of terms in this environment.

Technique. The LSI method in [11] uses Singular Value Decomposition (SVD), which belongs to the class of matrix decomposition techniques in linear algebra. To begin, we create a matrix where the rows represent the terms and its columns represent the documents. An element of the matrix represents the frequency of a term in a document. SVD expresses this matrix X as the product of 3 matrices, T, S and D^t, where S is a diagonal matrix of singular values ordered in descending order, and T and D are the matrices of eigenvectors of the square symmetric matrices XX^t and X^tX respectively. Furthermore, the square of the singular values are the eigenvalues for both XX^t and X^tX. The dimension of X is t (number of terms) by d (number of documents), while that of T is t by m, where m is the rank of X and is at most the minimum of t and d. S is an m by m matrix. Intuitively, SVD transforms the documents (columns of X) and the terms (rows of X) into a common space referred to as the *factor space*. The singular values in S are weights that are applied to scale the orthogonal, unit-length columns vectors of T and D and determine where the corresponding term or document is placed in the factor space. Similarity between documents or the likelihood of finding a term in a document can be estimated by computing distances between the coordinates of the corresponding terms and documents in this factor space.

The eigenvectors corresponding to the highest eigenvalues represent principal components that capture the most important characteristics of the data. The contributions keep diminishing for descending eigenvalues. By dropping some of the lower eigenvalues and corresponding eigenvectors, we lose some information, but can reduce the dimensionality of the data. This is useful when the number of documents is very large. We can retain the k highest eigenvalues, and the corresponding eigenvectors in the T and D matrices. The product $T_{txk}\, S_{\underline{kxk}}\, D^t_{kxd}$ gives the unique matrix of rank k closest to X based on a least-square distance metric. LSI is the process of using this matrix of lower rank to answer similarity queries such as which terms are strongly related and given

query terms, and what are the related documents. LSI has been shown to return query matches with higher precision when synonyms or multiple word senses would have prevented syntactic matching.

Experiment. We use LSI on the term by document matrix obtained from our document set. The terms were single words, bigrams and trigrams. We filtered out stopwords and the long tail of words that occurred only once. We reduced the rank of the matrix to k chosen such that 99% of the sum of squares of the singular values, which is the sum of eigenvalues, is retained. Next, we take the product $T_{txk}\, S_{kxk}$ which consists of the eigenvectors weighted by their singular values. In [17], Deerwester et.al. show that the comparison of two terms can be done in the factor space by taking inner product of corresponding rows in $T_{txk}\, S_{kxk}$. However, we want to extract the important terms. So we take the maximum absolute value in each row as the importance of that term, and sort based on this metric to order the terms by their descending importance. Given a threshold, our algorithm outputs all terms for which the metric, normalized to its maximum value, exceeds this threshold. When a new requirement document arrives in our system, LSI allows us to map it into a vector in the factor space by a simple matrix multiplication, and extract its important terms using this threshold technique.

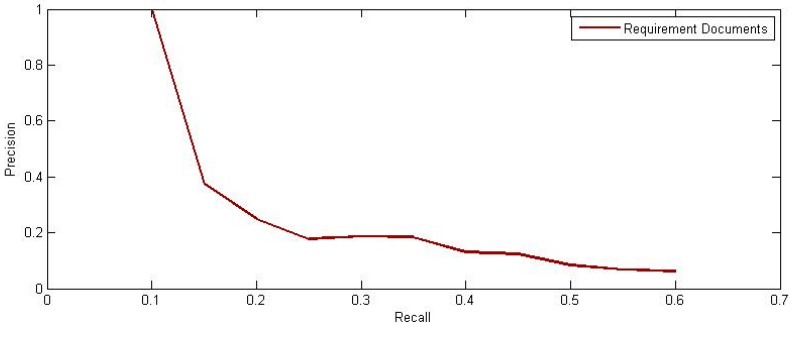


Fig. 1. Variation of Precision with Recall as threshold is varied

Since we need the ground truth to determine how well this approach works, we asked a human participant in our experiment to read the documents and identify important terms. To eliminate bias, this person had no a priori knowledge of the terms selected by our automated approach based on SVD and limited himself to manual analysis of the documents. He created a list of the top 20 important terms, henceforth referred to as the *ground truth* for this experiment. We started with the top 20 terms from the sorted list created by SVD, and progressively relaxed the threshold. At each stage, we calculated *precision* as the fraction of SVD's output that is present in the ground truth. We also calculated *recall* as the fraction of the ground truth that is present in the output of SVD. Our recall suffers due to our strategy of including all 2 and 3 letter words as stopwords. Thus the word 'ink', which is included in the ground truth, is absent from our term by document matrix. The same is true for terms such as "80# gloss" since we did not process any token such as "80#" that does not have at

least one alpha character. Figure 1 shows the variation of precision with recall as the threshold is varied. Precision drops from 1 to 0.25 when we relax our threshold progressively from 1 to 0.28. During this period, we observed only a small increase in recall to 0.2. The threshold was then relaxed progressively to 0.07. During this period, recall increased to 0.6, while precision dropped to 0.06.

6.2 Finding Service Candidates

We use a search engine (in particular, AltaVista) to find a pool of service providers through which the customer can engage in a business interaction over the web to meet its service needs. We use phrases such as "telemarketing service", "printing service" or "copyright litigation service" as input to retrieve the service provider pages.

In order to populate our data set (of service providers), we randomly selected words under the Wikipedia descriptions of the three services (telemarketing, printing and copyright litigation), and used them as input phrases to the AltaVista search engine. The choice is based on the fact that we were successful in accessing it programmatically. Recently, we have been made aware of BOSS [18], which is Yahoo!'s open search web services platform. We intend to redo our experiments using this API. Our understanding is that Google has stopped issuing new keys for its SOAP API that would have allowed programmatic search. Instead the new AJAX API must be used. It is geared towards browsers that need to tweak the display of search results, and will most likely not be suitable for our needs.

The pages retrieved by the search engine needed to be filtered as we seek only web forms and only web forms that contain the properties and attributes of the offered services to initiate a business engagement. Thus, for each retrieved web page by AltaVista, we retrieved the HTML source of the page to filter the non-form pages (or non-form sections of the form pages). We used the standard HTML tags that denote HTML form tags in order to filter out the non-form pages or non-form sections.

6.3 Identify Service Properties of Service Candidates

Once the pool of service providers (or, alternatively, the pool of forms, since there is a one-to-one mapping between forms and service providers) is determined by the methods of section 6.2, we seek to find the properties of each service type represented in the pool. For the experiments we conducted, there are three service types: telemarketing, printing and copyright litigation.

Throughout the remainder of the discussion, we denote each service type by m, each service providers (or form) by n, and each word used in the forms by w. We denote the number of service types, number of service providers, and the number of distinct words used in the forms by M, N and W, respectively. We note that the parameters N and W are pre-determined, since the pool of forms is already discovered in section 6.2. We assume that the service types m (and consequently the number of service types M) are known.

We use statistical learning techniques to identify properties of services based on the pool of service providers (or forms) retrieved by the methods discussed in section 6.2. We employ both supervised and unsupervised learning techniques. In supervised learning, the service type of each service provider n is assumed to be known, while in unsupervised learning this information is missing.

Data representation: We model each service provider (or form) *n* by a sequence of W bits (sequence of 0's and 1's), where each bit represents whether a word is present in the form. If the bit is 0, the word is absent, and if it is 1, the word is present.

Unsupervised learning: Clustering is an unsupervised learning technique, where objects that are close under some distortion criterion are clustered in the same group.

Supervised learning: When the cluster labels of the objects are already available (i.e., the service type of each service provider is known), one can use a supervised learning technique. We model the data as a mixture of M (where M=3) W-dimensional Gaussians, and estimated the parameters each Gaussian using the sample averages of the forms in that cluster. We use the k-means clustering algorithm for supervised classification with the squared-error distortion measure to cluster forms into M=3 groups. Each object to be clustered is a vector of 0's and 1's of length W, representing a form.

In total, 600 pages have been classified for this experiment for the three service categories with ~200 for each category. 122 of those pages have been form pages.

Results: We did not obtain encouraging results through unsupervised learning; however, supervised learning led to keywords that describe each service reasonably well. These words were selected based upon the probability of 50% or greater that the word will be found in all documents in the cluster. Keywords are shown in Table 1.

Table 1. Keywords describing service properties for three service categories

Service category	Significant keywords
Telemarketing	inbound, outbound, call, center, **telemarketing**, outsourcing, marketing, company, phone, address, zip, list.
Printing	business, card, **printing**, format, color, folder, sample, presentation, text, quote.
Litigation	intellectual, property, dispute, client, copyright, **litigation**, client, business, infringement, attorney, law, trial, website, competitor.

6.4 Rank Service Candidates

Input. In this step, we start with web pages of services identified from the web similar to step 6.2.

Objective. We seek an automated method, not based on domain-specific knowledge, which can identify the subset of the input services that match the required service properties of one of the requirement documents described in section 6.1. Since multiple matches are expected, a rank-ordered list of the services should be produced.

Technique. Singular Value Decomposition (SVD) was introduced in Section 6.1. We use SVD to index the service descriptions and treat one of the requirement documents as a term vector with which a query is performed.

Experiment. We use SVD on the term by document matrix obtained from the service web pages treated as documents. The HTML tags were removed. Again, we use

single words, bigrams and trigrams as the terms. We reduced the rank of the matrix to *k* chosen such that 99% of the sum of squares of the singular values, which is the sum of eigenvalues, is retained. The term vector for the query is transformed into the factor space obtained by SVD. This involves multiplying the transpose of the term vector by the SVD term matrix T_{txk}. The coordinates of the transform may be compared in this space to the other documents representing the services by accessing individual rows of the matrix product $D_{dxk}\, S_{kxk}$. For each row of this matrix product, we compute the inner product with the transform of the query term vector. Then we compensate for variable document sizes by normalizing the result by the product of the Euclidean length of the two vectors. The result is their *cosine similarity*, a standard measure for quantifying similarity of two documents in a vector space.

For our data, the human participant again established the ground truth without a priori knowledge of how SVD ordered the services. He did a manual evaluation of the requirement document used as query term vector. Based on his qualitative judgment, he provided us with a rank ordering of the services documents (HTML pages) in terms of how well they will meet the requirements specified in the query document.

We ranked the services based on cosine similarity of the service documents to the requirement document used for query. The correlation of two rank orders can be quantified by the Spearman rank correlation coefficient, which ranges between +1 when both rank orders are identical to -1 when one rank order is 1, 2,3,…,n and the other one is n,n-1,n-2,…,1. The results are presented in Table 2. In the absence of shared ranks, the Spearman coefficient is simply $1 - (6\sum d^2 / n\,(n^2 - 1))$. From this table, $\sum d^2$ is 106 and n = 17. So the Spearman coefficient is 0.87 for our experiment, indicating very high correlation.

Table 2. Comparison of service ranks obtained manually and through SVD for three service categories: printing (prt), telemarketing (tlm) and legal services (lwr)

Anonymized Service Names	Cosine Similarity	Cosine Rank	Manual Rank	d-squared
pitn-prt	0.4848	7	1	36
mpit-prt	0.7469	2	2	0
pitl-prt	0.6404	3	3	0
ppit-prt	0.8149	1	4	9
pitu-prt	0.5275	5	5	0
mkid-mlg	0.51	6	6	0
jsad-mlg	0.5665	4	7	9
pitn-mlg	0.4564	9	8	1
byro-tlm	0.2363	12	9	9
bbec-tlm	0.182	13	10	9
gtqo-tlm	0.3634	10	11	1
vnes-tlm	0.4821	8	12	16
spra-tlm	0.3045	11	13	4
il_o-lwr	0.0505	17	14	9
lwno-lwr	0.1524	14	15	1
cbrr-lwr	0.134	15	16	1
cmue-lwr	0.0709	16	17	1

In Table 2 we compare the service ranks obtained manually and through SVD for three service categories: printing, telemarketing and legal services.

This is further analyzed in Figure 2 where we plot the precision versus recall, assuming that the top 10 services in the manual rank column are relevant to the needs expressed in the requirement document. This is likely to be over-optimistic since an enterprise is likely to take the list of 17 services and put them through a qualification process and create a shortlist of vendors to whom requirements will be sent in future as part of a request for quote. Ideally, that shortlist should be the ground truth for calculating precision and recall. We assume that the top 10 services in the manual rank column will be fairly representative of the shortlist that an enterprise may generate if provided with these 17 services in response to requirement documents that are related to marketing campaigns. We observe from this graph that we can obtain a recall of 0.7 without losing any precision.

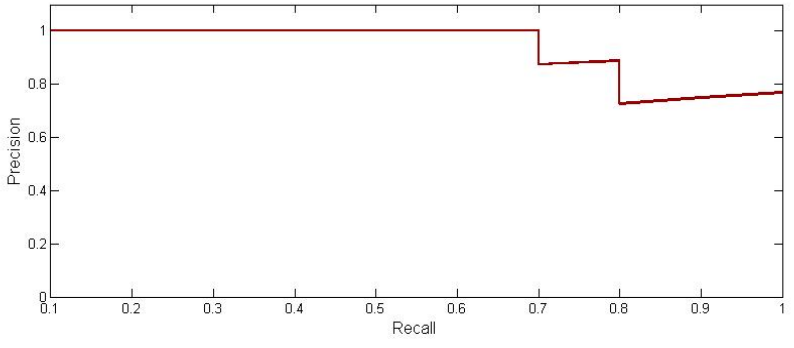


Fig. 2. Precision versus recall for ground truth consisting of top 10 services from manual ranking

6.5 Discussion

We consider the chosen techniques and experiments as an initial set which led to an initial set of answers. There is no doubt that the service selection problem remains difficult in an environment with largely unstructured data.

From the experiments in the first part, we can conclude that SVD provides a reasonable method for extracting important terms. As we have stated at the beginning of Section 5, our current goal was not to find the best method for each part, but to validate the approach. Since some of the terms provided as ground truth are 3-letter words, we conclude our policy of eliminating all 3-letter words as stopwords should be replaced by a more specific one based on lexical databases like WordNet.

For the second part, the identification of service providers from content returned from search using form tags on the web site, we can conclude that search engines such as Google, Yahoo and AltaVista retrieve not only the form pages that typically include attributes/capabilities for the services, but all pages that are related to the service being searched. Using the standard HTML tags that denote forms in order to filter out the non-form pages or non-form sections is a simple, but powerful technique in filtering out irrelevant text.

For the third part, the identification of service properties from form tags, we can conclude that we do not have a quantitative measure of accuracy. We note that the words in Table 2 reflect the major attributes of the services to which they correspond. For instance, for telemarketing services, the key attributes include the call volumes as well as numbers of inbound and outbound calls per minute. These are extracted by our algorithm and included as inbound, outbound and call. The keywords extracted for printing and copyright litigation also include major attributes of those services. We should point out, however, that not all keywords for each service have been extracted. For instance, for printing services, keywords that describe products to be printed (e.g., brochures, posters, etc.) are not extracted.

For the fourth part, the ranking of service provider candidates against a set of requirements using the SVD method, we compared against manual ranking and obtained high positive correlation. We can conclude that SVD was an effective means of ranking the services in the small set used for this experiment.

7 Conclusion and Future Work

The paper has presented an approach to the service selection problem using unstructured data for service properties. The problem has been addressed under the assumption that no common description framework exists. Four essential steps have been identified to address the problem. A search engine was employed to find service candidates from dense requirement formulations extracted from user input. A text classification technique was used to identify services and service properties from web content retrieved from returned search links. Service candidates were then ranked based on how well they supported desired properties using a Single Value Decomposition method.

An initial set of experiments has been conducted using a sampling of real requirements documents of an internal marketing department of a large corporation to procure external services for marketing campaigns. Descriptions of real service providers found on the web were used to extract service capabilities. Initial experiments showed encouraging results, but also exposed shortcomings that need to be addressed by further research.

A number of challenges have been uncovered by this research. We will address some of them in future work. One direction is to improve the techniques used for the service selection problem. Another direction aims at expanding from selecting services to also engaging them.

For the first direction, we will explore and develop stronger statistical learning techniques to improve accuracy for identifying service providers and extracting their properties from web content. We are also looking at incorporating structured data sources available on the web such as service catalogs or clustered information sources to improve results. The size of data sets (numbers of documents and services) we have considered was relatively small (10's of documents, 100's of web page from services). We need to explore how well the techniques scale over larger numbers of documents and services.

The second direction is to also actually engage services if they provide means of online engagement. It will open a whole new set of problems that needs to be

explored such as what different types of engagements can exist, how business relationships and service contracts are represented, and how engagement stages, steps and processes are described and executed, again under the assumption that no common formally defined framework can be assumed such as, for instance, RosettaNet's Partner Interface Processes (PIP) or Trading Partner Implementation Requirements (TPIR) [16] definitions or conventional Business Processes.

Acknowledgement

We would like to acknowledge the advice of Hamid Motahari for bringing the paper into its final shape.

References

1. Krafzig, D., Banke, K., Slama, D.: Enterprise SOA: Service Oriented Architecture Best Practices. Prentice-Hall, Englewood Cliffs (2005)
2. Universal Description, Discovery and Integration (UDDI), http://uddi.xml.org
3. OASIS ebXML Registry, http://www.oasis-open.org/committees/regrep
4. Lapata, M., Keller, F.: Web-based Models for Natural Language Processing. ACM Transactions of Speech and Language Processing 2(1), 1–30 (2005), http://homepages.inf.ed.ac.uk/mlap/Papers/tslp05.pdf
5. Powerset. Discover Facts. Unlock Meaning. Scan Summaries, http://www.powerset.com
6. W3C Semantic Web, http://www.w3.org/2001/sw
7. Chakrabarti, S.: Mining the Web: Analysis of Hypertext and Semi Structured Data. Morgan Kaufmann, San Francisco (2002)
8. Liu, B.: Web Data Mining: Exploring Hyperlinks, Contents and Usage Data. Springer, Heidelberg (2007)
9. Nasraoui, O., Zaïane, O.R., Spiliopoulou, M., Mobasher, B., Masand, B., Yu, P.S. (eds.) WebKDD 2005. LNCS (LNAI), vol. 4198. Springer, Heidelberg (2006)
10. WordNet. Cognitive Science Laboratory. Princeton University, http://wordnet.princeton.edu
11. Dumais, S.T., Furnas, G.W., Landauer, T.K., Deerwester, S.: Using latent semantic analysis to improve information retrieval. In: Proceedings of CHI 1988: Conference on Human Factors in Computing, pp. 281–285. ACM, New York (1988)
12. Hoovers. Online business registry, http://www.hoovers.com
13. Dun and Bradstreet. Provider of international and US business credit information and credit reports, http://www.dnb.com
14. ThomasNet. Provider of information about Industrial Manufacturers, http://www.thomasnet.com
15. Forman, G.: An Extensive Empirical Study of Feature Selection Metrics for Text Classification. Journal of Machine Learning Research 3, 1289–1305 (2003)
16. RosettaNet: Trading Partner Implementation Requirements (TPIR) Partner Interface Process (PIP) Maintenance Service, http://www.rosettanet.org/shop/store/catalogs/publications.html

17. Deerwester, S., Dumais, S.T., Landauer, T.K., Furnas, G.W., Harshman, R.A.: Indexing by latent semantic analysis. Journal of the Society for Information Science 41(6), 391–407 (1990)
18. Yahoo! Search BOSS (Build your Own Search Service), http://developer.yahoo.com/search/boss/
19. W3C, Web Services Description Language (WSDL) 1.1, http://www.w3.org/TR/wsdl
20. Dong, X., Halevy, A., Madhavan, J., Nemes, E., Zhang, J.: Similarity search for web services. In: Proceedings of VLDB 2004, pp. 372–383 (2004)
21. Ma, J., Zhang, Y., He, J.: Efficiently finding web services using a clustering semantic approach. In: Proceedings of WWW 2008 (2008)
22. Wang, Y., Stroulia, E.: Flexible Interface Matching for Web-Service Discovery. In: Proceedings of the Fourth international Conference on Web information Systems Engineering (2003)
23. XMethods, http://www.xmethods.com
24. Seekda, http://seekda.com/

On the Feasibility of Bilaterally Agreed Accounting of Resource Consumption

Carlos Molina-Jimenez, Nick Cook, and Santosh Shrivastava

School of Computing Science, Newcastle University, UK
{Carlos.Molina,Nick.Cook,Santosh.Shrivastava}@ncl.ac.uk

Abstract. The services offered by Internet Data Centers involve the provision of storage, bandwidth and computational resources. A common business model is to charge consumers on a pay-per-use basis where they periodically pay for the resources they have consumed (as opposed to a fixed charge for service provision). The pay-per-use model raises the question of how to measure resource consumption. Currently, a widely used accounting mechanism is provider-side accounting where the provider unilaterally measures the consumer's resource consumption and presents the latter with a bill. A serious limitation of this approach is that it does not offer the consumer sufficient means of performing reasonableness checks to verify that the provider is not accidentally or maliciously overcharging. To address the problem the paper develops bilateral accounting models where both consumer and provider independently measure resource consumption, verify the equity of the accounting process and try to resolve potential conflicts emerging from the independently produced results. The paper discusses the technical issues involved in bilateral accounting.

Keywords: Service provisioning, resource accounting, unilateral resource accounting, bilateral resource accounting, storage accounting, trusted outcomes, Amazon's storage service.

1 Introduction

The focus of our research is services provided by Internet Data Centers (IDC) to remote customers over the Internet. The variety of these services is large and still growing. Leaving aside specific technical details, we consider there to be three basic services that sell storage, bandwidth and compute power to remote consumers. We are interested in pay-per-use services, as opposed to fixed charge services. In the latter, the bill is fixed irrespective of the amount of resources consumed. In the former, the bill depends on the amount of resources consumed. Pay-per-use services can be further categorised into on-demand and utility services. In the case of on-demand services, the consumer pays (normally in advance) for a fixed amount of resource (for example, 60 minutes of international phone calls) and the service is terminated when the consumer exhausts the resources. With utility services the consumer consumes as much as he needs; charges are calculated according to actual consumption and presented to the consumer at the end of an agreed upon accounting period. A well

G. Feuerlicht and W. Lamersdorf (Eds.): ICSOC 2008, LNCS 5472, pp. 270–283, 2009.

known example from the IT field that uses the utility service model is the Amazon Simple Storage Service (Amazon's S3) [1] that sells storage space to remote users. Central to the pay-per-use model is the issue of accountability for the consumed resources: who performs the measurement and decides how much resource has been consumed — the provider, the consumer, a trusted third party (TTP), or some combination of them? Traditional utility providers such as water, gas and electricity services use provider-side accounting based on metering devices that have a certain degree of tamper-resistance and are deployed in the consumer's premises. Provider-side accounting is also widely used by phone services and Internet-based utility providers like Amazon. However, in contrast to traditional utilities, the infrastructure responsible for measuring resource consumption is deployed at the provider. The distinguishing feature of provider-side accounting is that it is unilateral. Provider-side accounting is acceptable when the consumer has good reasons to trust the provider not to accidentally or maliciously overcharge. However, we contend that there will be a class of applications, or of relationships between consumer and provider, where this assumption does not hold and where other models are needed. Unilateral consumer-side accounting can be implemented but we do not discuss it in this paper because it is not representative of practical applications — it is very unlikely that a provider would simply accept accounts unilaterally computed by a consumer. Unilateral accounting by a TTP on behalf of both consumer and provider would be more practical than consumer-side accounting, but in this paper we consider a hitherto unexplored alternative of bilateral accounting of resource consumption. We develop a new model in which the consumer and provider independently measure resource consumption, compare their outcomes and agree on a mutually trusted outcome. The problem of achieving mutual trust in accounting for resource consumption is currently neglected but is becoming important as users increasingly rely on utility (or cloud) computing for their needs. We explore the technical issues in bilateral accounting and develop a model that is abstract and general enough to apply to the different types of resources that are being offered on a utility basis. To ground our approach in current practice, we often use storage consumption as an example and, in particular, use Amazon's S3 as a case study for bilateral accounting.

Section 2 presents an overview of a generic accounting system, its components and underlying trust assumptions. Section 3 discusses our model of bilateral accounting for resource consumption and issues such as data collection and conflict resolution. In Section 4 we take Amazon's S3 storage service as a case study and discuss the feasibility of bilateral accounting of storage consumption. Section 5 presents related work. Section 6 concludes the paper with hints on future research directions.

2 Resource Accounting Services

We conceive a resource accounting system as composed out of three basic component services: metering, accounting and billing (see Fig. 1). We assume that resources are exposed as services through one or more service interfaces. As shown in the figure, the metering service collects data on resource usage at the service interfaces that it is able to access. The metering service stores the collected data for use by the accounting service. The accounting service retrieves the metering data, computes

resource consumption from the data and generates accounting data that is needed by the billing service to calculate the billing data. We assume that billing and accounting services use deterministic algorithms known to all the interested parties. Thus, given the same data, any party can compute the outcome of the relevant accounting or billing service.

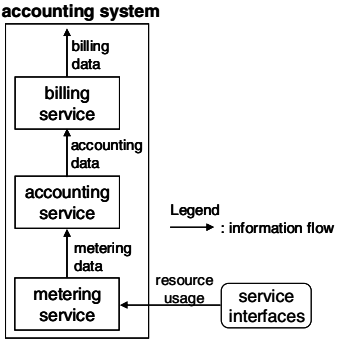


Fig. 1. Components of the resource accounting system

2.1 Trust Assumptions and Root of Trust

Regarding the trustworthiness of outcomes produced by the component services, we distinguish between **unilaterally trusted** and **mutually trusted** outcomes. A unilaterally trusted outcome is produced by a party with the help of its own component services and is not necessarily trusted by other parties. The components could be located either within or outside the party's infrastructure. In the latter case, the component's owner may need to take additional measures, such as the use of tamper-resistant mechanisms, to protect the component and its outcomes against modification by other parties [2, 3]. There are two approaches to producing a mutually trusted outcome: 1) a TTP produces the outcome using its own certified infrastructure, or 2) the parties concerned use their respective unilaterally trusted outcomes as the basis for agreement on a valid, mutually trusted outcome. This alternative is the focus of our interest and, as discussed later, requires the execution of some protocol between the participants (two in a bilateral relationship) to produce an agreed upon and non-repudiable outcome. Mutually trusted outcomes form the "root of trust" for building trusted resource accounting systems. The source of mutually trusted outcomes can be rooted at any one of the three levels shown in Fig. 1. Once a mutually trusted outcome source is available, the trustworthiness of the component services above it becomes irrelevant. Given the determinacy assumption, a party can always resort to the mutually trusted outcome to compute and verify results produced by other parties. For example, given a metering service that produces mutually trusted metering data, the accounting and billing services can be provided by any of the parties in any combination; their outcomes are verifiable by any other party.

3 Bilateral Resource Accounting

Bilateral accounting is an attractive solution in applications where mutually untrusted consumer and provider are reluctant, or unable, to use a TTP and therefore agree to deploy their own component services. A distinguishing feature of this approach is that the consumer and provider own and run their own independent but functionally equivalent component services to produce their unilaterally trusted outcomes. Bilateral agreement between the pair of component services results in the trusted outcome needed to build the whole resource accounting system. This approach presents us with two fundamental problems: (i) how do the consumer and provider collect the metering data that is essential to compute unilaterally trusted outcomes that can form the basis for agreement on resource consumption, and (ii) how do consumer and provider resolve conflicts over resource consumption.

3.1 Collection of Metering Data

As stated in Section 2, we assume that a resource user (consumer or provider) uses a resource by invoking operations at service interfaces to the resource. Consumers use a resource through one or more service interfaces exposed by the provider. In their turn, the provider accesses the resources necessary to provide the service through a set of interfaces. The consequence is that any party wishing to collect metering data is restricted to analysing operations at the service interfaces that they can access.

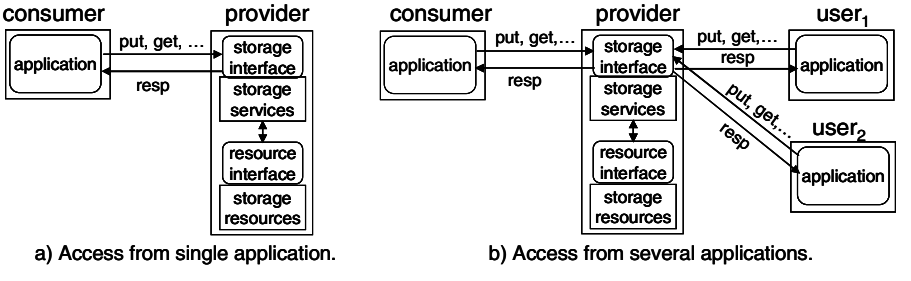


Fig. 2. Example use cases for utility storage services

For example, in Fig. 2-a, the consumer accesses the provider's storage services through a storage interface. To generate its unilaterally trusted outcome with respect to storage consumption, the consumer must generate metering data by analysing operations at this storage service interface. In contrast, the storage provider has access both to the storage service interface it exposes to the consumer and to a resource interface for operations on its storage resources. Consequently, it can collect metering data by analysing operations at both interfaces, combine this information and produce its own unilaterally trusted outcome about the consumer's resource consumption. For the sake of illustration, we consider two common utility storage use cases. In Fig. 2-a, a consumer hosts a single application that invokes operations on the provider's storage service. An example of this case is the use of a storage service for backup. In Fig. 2-b, the storage is consumed by the consumer's application and by applications

hosted by other users ($user_1$, $user_2$, etc.) that all access the storage service at the consumer's expense. An example of this case is a consumer using a storage service to provide photo or video sharing services to other users.

In the single application case, the consumer can analyse its requests to and the responses from the service over a given period. The consumer can collect metering data on the type of a request (put, get etc.), on the size of a request and of an associated response, and on the round-trip time for request/response pairs. It can compute the bandwidth used for its service invocations (for example, in GB transferred per month). Given a defined relationship between data transferred and amount of storage used (see Section 4), from request types and request/response sizes the consumer can compute storage consumption (for example, in GB stored per month). From round-trip times, it can compute average response times for its service invocations. On the other hand, the provider can compute bandwidth usage by performing essentially the same analysis as the consumer on the request/response traffic as seen by the provider. It can use the same data to compute storage consumption in the same way as the consumer. In addition, the provider is able to analyse operations at the storage resource interface. That is, the provider can compute storage consumption based on data collected at either the storage service interface or at the resource interface (which may allow a more direct measure of consumption). The choice of input data will determine the algorithm the provider uses to compute its unilaterally trusted outcome. Equally important, the choice of input data admits the possibility of a degree of divergence between the unilaterally trusted outcomes produced by consumer and provider. With respect to response time as experienced by the consumer, the provider must either: (i) compute estimates based on average network latency, or (ii) compute actual response times using data the provider collects at the consumer. The former approach can be used if the provider-side data supports computation of response times to a degree of accuracy that is acceptable to both parties. The latter approach can be used if the provider can deploy their own metering service at the consumer.

The multi-user, multi-application case shown in Fig. 2-b presents the consumer wishing to independently compute resource usage with additional problems. The applications all access the resource through the storage service interface and the consumer is accountable for all usage. However, in addition to collecting data on its own operations, the consumer must now collect data on operations performed by $user_1$ and $user_2$ (and any additional users given access). The consumer may be able to collect this data by deploying metering services with each application instance, in their own and each user's infrastructure. An advantage of this approach is that the consumer can collect data to compute bandwidth consumption, storage consumption and response times for all the applications for which it is accountable. The feasibility of the approach will depend on factors such as the degree of control the consumer has over the deployed applications and the number of users. Alternatively, the consumer may be able to deploy a metering service at the provider to collect data on user operations at the storage service interface. If this approach is adopted, the consumer can compute bandwidth and storage consumption (under the same assumptions as the single application case). They can no longer directly compute response times for any party other than themselves. As for the provider in the single application case, estimates of response times, to some agreed degree of accuracy, may be acceptable.

From the storage provider's viewpoint, the multi-user, multi-application case is similar to the single consumer application case. The provider can compute resource consumption such as bandwidth and storage from data collected at the storage service and/or resource interfaces. The discussion of provider computation of consumer-perceived response times applies to computation of response time for other users.

The preceding discussion highlights the fact that bilateral accounting relies on the ability of the parties involved to independently collect the data necessary to produce their unilaterally trusted outcomes. Further, as noted in Section 2, data collection may require additional protection mechanisms such as tamper-resistance. We now generalise the discussion.

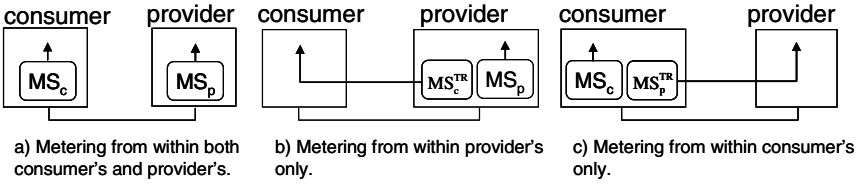


Fig. 3. Models for bilateral metering

Fig. 3 shows three possible scenarios. A provider (*p*) offers some service to a consumer (*c*). MS stands for Metering Service. MS_c is a consumer's metering service. MS_P is a provider's metering service. *TR* means tamper-resistant protection. So, MS_c^{TR} is a consumer metering service that is tamper-resistant to provider modification. The outcomes of metering services are unilaterally trusted (MS_c by c and MS_p by p) and made available to their respective accounting services (not shown). Fig. 3-a applies when metering data can be collected from within the consumer and the provider (for example, data to compute bandwidth). Fig. 3-b applies when the metering data of a required degree of accuracy can only be collected from within the provider (for example, data on access by multiple users). In this case, the provider deploys its MS_p locally, whereas the consumer performs remote metering with the help of MS_c^{TR}. Fig. 3-c mirrors Fig. 3-b: metering data of a required degree of accuracy can only be collected from within the consumer (for example, data to compute response time). In this case, MS_c performs local metering whereas the provider deploys MS_p^{TR} at the consumer.

3.2 Agreement on Mutually Trusted Accounting Outcomes

Though they are functionally equivalent, consumer and provider components do not necessarily use the same algorithms or input data to compute their unilaterally trusted outcomes. As discussed in Section 3.1, they may use data collected at different interfaces to compute an outcome. There is then the possibility of divergence between the independently computed (unilaterally trusted) outcomes. To address this problem we suggest the use of a Comparison and Conflict Resolution Protocol (CCRP). A suitable protocol will support: (i) the comparison of independently produced and unilaterally trusted outcomes to detect potential divergences; (ii) where possible, for

example, when $|Outcome_p - Outcome_c| \leq d$, ($c$, p, d stand for consumer, provider and agreed-upon acceptable divergence, respectively), the immediate declaration of absence of conflicts; (iii) where the divergence is greater than *d,* the execution of a negotiation protocol between consumer and provider with the intention of reaching agreement on a single outcome; (iv) when the negotiation protocol fails to sort out the conflict automatically, the declaration of conflict for off-line resolution; (v) production of non-repudiable mutually trusted outcome. The non-repudiation property is necessary to ensure that neither consumer nor provider can subsequently deny execution of the CCRP or the result of that execution. That is, they could not subsequently deny their agreement or otherwise to a given accounting outcome.

Our middleware for non-repudiable information sharing [4, 5] can form the basis of a CCRP. The middleware provides multi-party, non-repudiable agreement to updates to shared information which can be maintained in a distributed manner with each party holding a copy. Essentially, one party proposes a new value for the state of some information and the other parties sharing the information subject the proposed value to application-specific validation. If all parties agree to the value, then the shared view of the information is updated accordingly. Otherwise, the shared view of the information remains in the state prior to proposal of the new value. For non-repudiable agreement to a change:

1. there must be evidence that any proposed change originated at its proposer, and
2. there must be evidence that all parties agreed to any change and therefore share the same (agreed) view of information state.

That is, there must be evidence that all parties received the proposed update and that they agreed to the state change.

In our consumer-provider application the problem is for consumer and provider to reach agreement on the set of mutually trusted outcomes that will ultimately be used to compute the bill for a given accounting period. For example, if bills are computed from a set of resource usage records for a given period, the problem is to reach agreement on the valid membership of the set. Fig. 4-a shows the abstraction of a consumer and provider sharing, and adding to, a set of resource usage records; the box is shown dotted to indicate that this is a logical view; in reality each party has a local copy of the set. The non-repudiable information sharing middleware will maintain this abstraction of a set of mutually agreed records and control the addition of new records to the set. The middleware generates non-repudiation evidence to ensure that the state of the agreed set of records is binding on both parties. In the case of the provider proposing a new record, the basic two-party agreement process is:

1. The provider proposes a new record.
2. The consumer performs application specific validation of the proposed record. This validation can be arbitrarily complex. The record could be compared with an equivalent record computed locally by the consumer. Bounds on divergence between the two records could be imposed. The history of previous interactions could be used to detect any persistent over-estimation of consumption on the part of the provider.
3. The consumer returns a decision on the validity, or otherwise, of the proposed record.

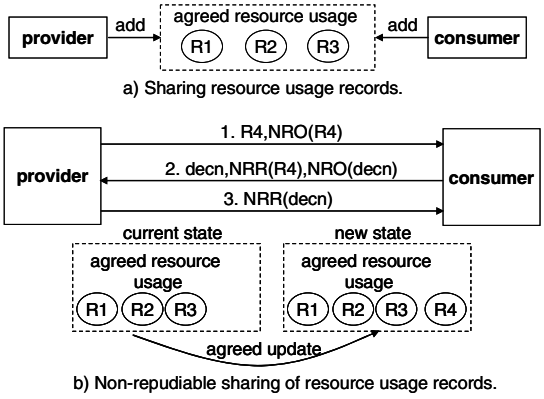


Fig. 4. Non-repudiable agreement to resource usage

The middleware uses a signed, two-phase commit protocol with application-level validation to achieve the preceding basic agreement. Fig. 4-b shows the execution of the protocol for addition of a new record, R4, to the agreed set of records {R1, R2, R3}. First, the provider proposes the addition of R4 with non-repudiation of its origin (NRO(R4)). Then, the consumer validates the record and returns a decision on its validity or otherwise (decn), non-repudiation of receipt of R4 (NRR(R4)) and non-repudiation of origin of the decision (NRO(decn)). The decision is essentially a binary yes or no value. However, the middleware supports annotation of the decision with application-specific information. For example, the decision may be annotated with the degree of divergence of the record from the consumer's view of resource usage. These annotations form part of the evidence generated during protocol execution and may be used to support more complex negotiation built on top of the basic agreement mechanism. The protocol terminates with the provider sending non-repudiation of receipt of the validation decision to the consumer (NRR(decn)). As shown in Fig. 4-b, if the consumer decides that R4 is valid then the agreed set of records is {R1, R2, R3, R4}. Otherwise, the agreed view of the set remains unchanged ({R1, R2, R3}) and the failure to agree to the addition of R4 will be signaled to both parties. As with annotations to decisions, this failure signal can be used to build more complex negotiations. For example, the signal could trigger the proposal of a revised record or the initiation of extra-protocol dispute resolution. At the end of a protocol run, both parties have the same irrefutable (binding) view of the set of agreed records and of the validation decisions made with respect to records in the set and with respect to proposed records that have been rejected by either consumer or provider.

As discussed, the non-repudiable agreement provided by the middleware can be used as a basic building block for negotiation of a mutually trusted accounting outcome. Proposal of, and agreement to, addition of a new outcome is cheap (involving a single agreement protocol round). Disagreement to the proposed addition of an outcome to the set in one round of protocol execution can trigger further protocol rounds with revised proposals or lead to extra protocol resolution (based on evidence generated during protocol executions). The middleware supports the application-specific, autonomous imposition of conditions for validation of updates,

proposal of revised updates and termination of protocol execution. Therefore, more complex multi-round negotiation could be built on the basic agreement mechanism.

3.3 Models for Bilateral Accounting

We now complete our models for bilateral accounting by combining the component services with execution of a CCRP. Fig. 5 shows the three variants of our model when metering data is collected at both consumer and provider (as in Fig. 3-a). As before, c and p stand for consumer and provider. MS, AS and BS stand for metering service, accounting service and billing service, respectively. The difference between the three variants is the level at which the root of trust is established. This is determined by the level at which the parties execute the CCRP to agree on a mutually trusted outcome.

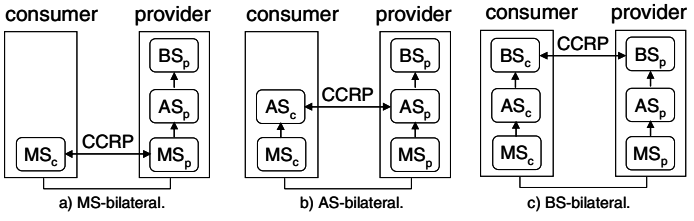


Fig. 5. Models for bilateral resource accounting

In Fig. 5-a the mutually trusted outcome is produced at the metering level, at the accounting level in Fig. 5-b and at the billing level in Fig. 5-c. In each variant, the provider deploys the component services above the level of the root of trust. As indicated in Section 2, the rational here is that once a bilaterally trusted outcome is available, the consumer can verify any subsequent accounting computation. This verification is orthogonal to the operation of the bilateral resource accounting system. There are six further variants of the model: three for deployment of MS_c and tamper-resistant MS_p at the consumer (as in Fig. 3-c), and three for deployment of MS_p and tamper-resistant MS_c at the provider (as in Fig. 3-b). The full set of models also allows for the combination of data from mixed MS deployments to build a complete bilateral accounting system.

4 On the Feasibility of Bilateral Accounting for Amazon's S3 Storage

Amazon's S3 is currently one of the best-known services that provides storage to remote users over the Internet on a pay-per-use basis. S3 presents its users with the abstraction of buckets that contain arbitrary data objects up to 5GB in size with 2KB of metadata. Each bucket is owned by an Amazon's S3 user. A user-assigned key identifies each object in a bucket. There are both REST-style and SOAP Web service interfaces to the service. The interfaces define operations to create, list, delete and retrieve buckets and objects. There are other operations to manage object access control and metadata and to obtain limited statistics of service usage. User requests to

perform operations on the storage service are authenticated by a user signature and signed timestamp. Amazon charges S3 consumers monthly for 1) the amount of GB stored, 2) the number of *put*, *get* and *list* requests generated by the consumer, and 3) the amount of bandwidth consumed by the consumer (bytes transferred from outside Amazon into S3 and in the opposite direction). At least twice daily Amazon checks their storage resources to calculate the amount of storage occupied by a consumer's buckets and multiplies this by the amount of time elapsed since the last check. Amazon charges a flat amount for every 1000 *put* or *list* request and a flat amount for every 10000 *get* request. There is no charge for *delete* request. Download (e.g. object retrieval) is approximately twice as expensive as upload (e.g. object creation). Currently, accounting for use of Amazon's S3 is provider-side. Amazon meters the consumer's consumption, calculates the charges and presents the consumer with a bill. According to the definitions in Section 2.1, Amazon's charges are based on unilaterally trusted outcomes. Ideally, resource consumers should have mechanisms to independently measure their resource consumption and verify that they are not accidentally or maliciously overcharged. This would result in a bilateral accounting system.

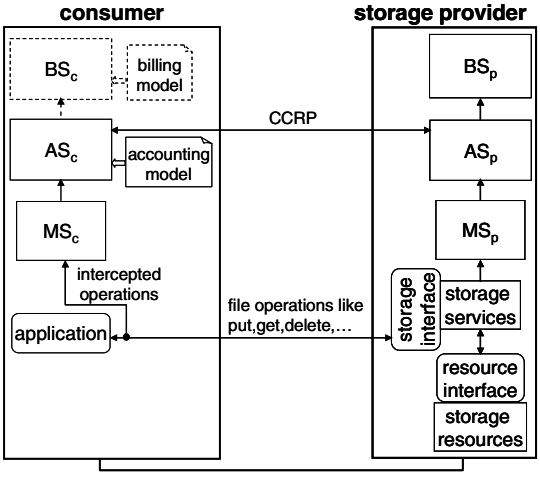


Fig. 6. Model for bilateral accounting of resource consumption in Amazon's S3

For the specific example of accounting for S3 storage consumed by an application deployed within the consumer's infrastructure (as in Fig. 2-a), we can envisage a bilateral accounting system based on the abstract model illustrated in Fig. 6. It is relatively easy to bilaterally meter the number and type of requests and bandwidth consumption. As discussed in Section 3, both consumer and provider can collect the necessary data at the storage service interface. In the case of Amazon, MS_c and MS_p can independently intercept a consumer's requests and provide the metering data to AS_c and AS_p. Given that MS_c and MS_p are intercepting the same request/response traffic, the two accounting services should arrive at the same result. Thus, it is straightforward to bilaterally account for two elements of Amazon's charging model:

numbers of requests and bandwidth. A more challenging problem is to ensure that AS_c and AS_p produce similar accounting data about storage consumption over time (the third element of Amazon's charging model). The difficulty with storage consumption is that the consumer does not have the same degree of access to the storage resources as Amazon. Amazon, as storage provider, accesses the resources though their resource interface. This allows Amazon to more directly measure resource consumption. Amazon's S3 does provide operations to list storage information, to obtain service statistics and perform some user-accessible logging. However, Amazon explicitly states that this information cannot be relied on for checking consumer's storage accounting. In any case, these operations cannot be used by MS_c as an independent means of collecting metering data about storage consumption. As suggested in Section 3.1, this leads to different strategies for collection of metering data by MS_c and MS_p. For example, MS_c can independently meter the size of upload requests to Amazon. AS_c can then use such consumer-side data to estimate storage consumption. However, it is very likely that the results produced by AS_c will diverge from those produced by AS_p because AS_p is able to rely on data collected by MS_p which has access to the resource interface within Amazon. There is clearly some relationship between the request size that a consumer can measure and the storage resource usage (which may for example include file meta data) that Amazon is able to measure more directly. Unfortunately, Amazon's published charging model does not define this relationship. Furthermore, the charging model does not provide sufficient information to the consumer about the time from which storage usage is charged. Thus, when computing the impact of a request to create or delete an object or bucket, it is unlikely that the consumer AS_c will arrive at the same result as the provider AS_p. In summary, the only independent metering data available to the consumer is based on their request traffic. The published charging does not provide them with sufficient information to use that data to perform their own storage accounting and produce results that will be compatible with those produced by Amazon. Bilateral accounting is currently possible for two elements of the Amazon's S3 charging model (number of request and bandwidth usage) but not for the third (storage usage over time).

A solution to the preceding problem is that the provider publish a reference model that allows the consumer to estimate with an agreed upon accuracy the impact of their requests on storage usage. In Fig. 6, this idea is represented by the provider-supplied accounting model that the consumer's accounting service uses to generate accounting data. For example, an Amazon's S3 accounting model for use by consumers would stipulate how to calculate the impact on storage usage of a put request given the time and size of the request. This would include the means to estimate the size of any storage metadata and the time from which Amazon would charge storage. Such a model is necessary but, given they are using different input data, AS_c and AS_p may still produce divergent unilaterally trusted outcomes. Therefore, as shown, in this case AS_c and AS_p would execute the CCRP to arrive at a mutually trusted outcome that becomes the root of trust for billing. In principle, the root of trust could be produced at the metering services or billing services level. However, in this example, reaching agreement at the metering services level is difficult because there is a significant difference between the data on storage usage collected by the consumer and provider. It is only when AS_c and AS_p apply the accounting model that outcomes will converge.

The availability of a mutually trusted outcome at the accounting level makes bilateral billing redundant. This is why we represented consumer's BS_c in dashed lines. The argument here is that the consumer can always retrieve the bilaterally trusted accounting outcome, input it into a billing service that implements the billing model and verify the bill.

5 Related Work

The topic of bilateral monitoring of resource consumption has received little attention as an object of study in its own right. In this section we will briefly mention some results that are somehow related to our work. General architectural concepts involved in monitoring service level agreements are discussed in [6]. The emphasis there is in providing the two parties with mechanisms for monitoring each other's behaviour. In [7], the authors introduce the concept of monitorability of service level agreement parameters and point out that some parameters are unmonitorable, monitorable by the consumer, monitorable by the provider or monitorable by a TTP. This relates directly to the discussion in Section 3.1 on the collection of data at the set service interfaces available to a given party. Regarding models for resource accounting, in [8] authors describe a reference model for building accounting services based on accounting policies. The layers of this model correspond to the component services of our work, however, the focus of the authors is only on unilateral accounting. To the best of our knowledge, Saksha [9] is one of the first attempts to implement bilateral accounting of storage consumption. The paper discusses only scenarios with a single application deployed within the consumer like in Fig. 2-a. The authors do not explicitly describe their trust model which appears to be a combination of what we call consumer-side and provide-side metering, where the consumer and the storage provider unilaterally measure the amount of storage consumed and freed, respectively. The merit of this work is that it raises several issues that need to be studied. Our decomposition into component services is similar to that suggested in [10] where the issue of storage accounting in Grid environments is addressed; the interest of the author is in unilateral accounting, whereas in our work, the focus is on bilateral. A general discussion on the use of strong accountability from the perspective of liability for actions (e.g., deletion of files, retrieval of the latest version, etc.) executed between a client and their storage service provider is presented in [11]; however, the issue of storage consumption accounting is not discussed. The need for bilateral accountability of storage consumption is hinted in [12] where a storage provider charges for executing their consumer's operations such as create, replace, append, delete, find and list file, etc, depending on the size of the file. The paper focuses on the payment mechanisms and overlooks the issue about the computation of the size of the file which seems to be unilaterally decided by the consumer. Comprehensive discussions of Amazon's storage and an application hosting services is presented in [13, 14]. Bilateral accounting of storage consumption is not mentioned, yet related parameters such as throughput, availability, consistency, etc. are discussed. Our work is related to research on the financial impact of IT solutions. In [15] for example, the authors study analytical models for resource allocation and job acceptance in a Web service hosting enterprise, that allow the enterprise to maximize its revenue. Bilateral accounting

opens up the possibility of empowering consumers to minimize their expenditure on IT resources. We believe that a consumer can take advantage of their metering data to infer expenditure pattern and tune the application to minimise the bill; a step further would be for the consumer to analyse accounting models offered by different providers with the intention of dynamically re-locating the application to the provider whose accounting model would minimize the expected resource consumption charges.

6 Concluding Remarks

Users are increasingly relying on pay-per-use services from utility (or cloud) computing service providers. Charging is on the basis of the provider unilaterally measuring the consumer's resource consumption and producing the accounting data. We believe that this practice of provider-side accounting will need to be supplemented by measures that enable consumers to produce their own accounting data, minimally to check the reasonableness of the provider produced data. Bilateral accounting of resource consumption is the next logical step: the consumer and the provider independently measure resource consumption, compare their outcomes and agree on a mutually trusted outcome. We developed a number of models of bilateral accounting and used Amazon's S3 storage services as a case study to highlight the issues involved. Success of bilateral accounting to a large extent will depend on two factors: the quality of accounting data consumers can collect and the availability of a relatively simple comparison and conflict resolution protocol (CCRP) to enable production of mutually agreed outcomes. Service providers can help consumers by providing (i) suitable service interfaces to enable consumer side metering, and (ii) a reference model (e.g., an accounting model) to enable consumers to estimate resource consumption and associated charges. Further, as we discussed, sometimes there is also a need for a consumer (provider) to collect metering data directly at the provider's (consumer's) premises, so suitable metering techniques will need to be developed. CCRP procedures will also need to be developed and agreed as a part of the service level agreement.

Acknowledgements

This work has been funded in part by UK Engineering and Physical Sciences Research Council (EPSRC), Platform Grant No. EP/D037743/1, "Networked Computing in Inter-organisation Settings".

References

1. Amazon Simple Storage Service (Amazon S3), `http://aws.amazon.com/s3`
2. van Oorschot, P.C.: Revisiting Software Protection. In: Boyd, C., Mao, W. (eds.) ISC 2003. LNCS, vol. 2851, pp. 1–13. Springer, Heidelberg (2003)
3. TCG Specification Architecture Overview. Specification Revision-1.4, `http://www.trustedcomputinggroup.org/`

4. Cook, N., Shrivastava, S.K., Wheater, S.: Distributed Object Middleware to Support Dependable Information Sharing Between Organisations. In: IEEE International Conference on Dependable Systems and Networks (DSN 2002), Washington DC, pp. 249–258. IEEE Computer Society, Los Alamitos (2002)
5. Cook, N., Robinson, P., Shrivastava, S.K.: Design and Implementation of Web Services Middleware to Support Fair Non-repudiable Interactions. Int. J. Cooperative Information Systems (IJCIS) Special Issue on Enterprise Distributed Computing 15(4), 565–597 (2006)
6. Molina-Jimenez, C., Shrivastava, S.K., Crowcroft, J., Gevros, P.: On the Monitoring of Service Level Agreements. In: First IEEE International Workshop on Electronic Contracting, San Diego, pp. 1–8. IEEE Computer Society, Los Alamitos (2004)
7. Skene, J., Skene, A., Crampton, J., Emmerich, W.: The monitorability of service-level agreements for application-service provision. In: Sixth International Workshop on Software and Performance (WOSP 2007), Buenos Aires, Argentina, pp. 3–14. ACM, New York (2007)
8. Zseby, T., Zander, S., Carle, G.: Policy–Based Accounting, IETF RFC 3334 (October 2002), http://www.ietf.org/rfc/rfc3334.txt
9. Kher, V., Kim, Y.: Building Trust in Storage Outsourcing: Secure Accounting of Utility Storage. In: 26th IEEE International Symposium on Reliable Distributed Systems (SRDS 2007), Beijing, pp. 55–64. IEEE Computer Society, Los Alamitos (2007)
10. Scibilia, F.: Accounting of Storage Resources in gLite Based Infrastructures. In: 16th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2007), Paris, pp. 273–278. IEEE Computer Society, Los Alamitos (2007)
11. Yumerefendi, A.R., Chase, J.S.: Strong Accountability for Network Storage. ACM Trans. on Storage 3(3) (2007)
12. Ioannidis, J., Ioannidis, S., Keromytis, A.D., Prevelakis, V.: Fileteller: Paying and Getting Paid for File Storage. In: Blaze, M. (ed.) FC 2002. LNCS, vol. 2357, pp. 282–289. Springer, Heidelberg (2003)
13. Garfinkel, L.S.: An Evaluation of Amazon's Grid Computing Services: EC2, S3 and SQS. Technical Report TR-08-07; Center for Research on Computing and Society, School of Engineering and Applied Science, Harvard University (2007)
14. Brantner, M., Florescu, D., Graf, D., Kossmann, D., Kraska, T.: Building a Database on S3. In: Annual ACM Conference (SIGMOD 2008), Vancouver, pp. 251–263. ACM, New York (2008)
15. Mazzucco, M., Mitrani, I., Palmer, J., Fisher, M., McKee, P.: Web Service Hosting and Revenue Maximization. In: Proceedings of the Fifth IEEE European Conference on Web Services (ECOWS 2007), Halle, Saale, Germany, pp. 45–54. IEEE Computer Society, Los Alamitos (2007)

On Analyzing Evolutionary Changes of Web Services*

Martin Treiber, Hong-Linh Truong, and Schahram Dustdar

VitaLab, Distributed Systems Group
Institute of Information Systems
Vienna University of Technology
{treiber,truong,dustdar}@infosys.tuwien.ac.at

Abstract. Web services evolve over their life time and change their behavior. In our work, we analyze Web service related changes and investigate interdependencies of Web service related changes. We classify changes of Web services for an analysis regarding causes and effects of such changes and utilize a dedicated Web service information model to capture the changes of Web services. We show how to put changes of Web services into an evolutionary context that allows us to develop a holistic perspective on Web services and their stakeholders in a ecosystem of Web services.

1 Introduction

Web services undergo changes during their life time. This behavior is of particular interest when putting Web services in the context of Web service ecosystems [1]. With regard to Web services, a Web service ecosystem is the environment in which a Web service operates. The environment consists of different stakeholders that have interest in Web services and influence or control the life cycle of a Web service. To understand these ecosystems, we need to understand Web services with regard to evolutionary aspects since they are the central entities in such ecosystems. In particular, we need to analyze the artifacts that have impact on Web services, Web service ecosystems respectively, and investigate the reasons for changes of Web services.

Currently there is little support for Web service evolution, even though the evolution of Web services accounts for major development costs. The evolution of Web services involves changes of requirements, implementation modifications, changes of Web service semantics, changes of Web service usage and so on. These activities originate from different stakeholders, such as developers, providers, brokers, end users and service integrators that interact in Web service ecosystems.

In this work, we focus on the complexity of evolutionary Web service modifications. It's important for Web service providers to understand the prerequisites and the consequences of modifications of Web services in the light of limited resources (time, manpower, money). Current practices to describe Web services do not take these dynamic aspects into account. Approaches such as (WSDL [2], OWL-S [3], WSMO [4], WSDL-S [5]) primarily focus on interface related issues and do not model changes. We

* The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement 215483 (S-Cube).

G. Feuerlicht and W. Lamersdorf (Eds.): ICSOC 2008, LNCS 5472, pp. 284–297, 2009.

argue that an evolutionary model requires a holistic perspective on Web services and needs to integrate information from the perspective of various stakeholders as well as different (design time and run time) data sources [6].

Real world Web services provide valuable input for our investigations with regard to the understanding of Web service evolution. In this paper we are using real world Web services of an Austrian SME called Wisur [1] to illustrate the challenges concerning Web service evolution. Wisur provides business related information to customers. Their main business is to sell business reports with information about companies (turnover, financial situation, etc) and consumer data (address, date of birth, etc.) to their customers. Wisur's customers use Wisur's Web services to search in Wisur's database and to access the desired information [2]. As soon as the requirements of customers change, Wisur needs to adapt their services accordingly. Examples are for instance the addition of new functions to existing services, or the creation of new Web services. In addition, Wisur reviews it's Web services once a month to look for issues that might cause problems in the future or did cause problems in the past (e.g., SLA violations). Future activities, such as a planned augmentation of a database with consumer data can lead to longer execution times of queries. This has impact on the overall execution time of the Web service that queries data from this database. When putting these activity into the context of Web service evolution, we can observe the need for classification of modification activities under the umbrella of Web service evolution. In this paper, we focus on the evolution of single Web services that we consider as atomic building blocks of Web service ecosystems. In particular we analyze changes of Web services based on empirical observations. We investigate the impact of these changes from different perspectives. We propose a methodology that identifies influencing factors of Web services and a model of Web service related changes. Our major contribution is to define the basic element of Web service evolution, i.e. the *evolutionary step*, and embed it in Web service ecosystems. The rest of the paper is organized as follows. After summarizing related work in Section 2 we provide our analysis of changes of Web services in Section 3. We conclude the paper with a discussion and an outlook in Section 4.

2 Related Work

From an organizational point of view, our work is related to service management in general. Approaches such as WSDM [7] or WSMF [8] offer frameworks to collect data about Web services and to manage them. The work of Casati et. al. [9] focuses on the business perspective of Web service management. Fang et. al. [10] discuss the management of service interface changes. The authors describe version-aware service descriptions and directory models. The work by Kaminski [11] introduces a framework that supports service interface adaptors for backward compatible versioning of Web services. Our work covers different aspects of service changes, such as runtime aspects (e.g., QoS), implementation changes, semantic changes, etc.

[1] http://www.wisur.at

[2] See http://webservice.wisur.at:8000/axis/services/WISIRISFuzzySearchService?wsdl for an example of a Web service.

The collection of run time information about Web services (usage statistics, logging, etc.) is discussed by Ghezzi et. al. [12]. The area of Software Configuration Management [13] is also related to our research with regard to analysis of changes of software systems and versioning issues. We follow ideas such as the collection, classification and monitoring of change events from the aforementioned approaches. However, our approach differs insofar, since we focus on evolutionary aspects that are not covered by management approaches.

From an evolutionary perspective, our work is closely related to evolutionary aspects of software in general. Of particular importance is the work of in [14] where the authors studied software changes and considered software systems as dynamic entities that change over time. Lessons can also be learned from the basic SPE classification scheme which is extensively discussed in [15] and [16].

Papazoglou [17] discusses the challenges of service evolution and introduces two categories of service related changes. While similar in spirit, our work follows a broader scope, since our approach introduces a concrete information model to capture service related changes and lays the foundation for deeper analysis of service related changes. Andrikopoulos et. al. [18] discuss the management of service specification evolution. The author's abstract service definition model addresses Web service evolution on an abstract layer whereas we follow a bottom up approach with the analysis of single Web services with regard to Web service evolution.

The work in [19] describes the use of a dependency structure matrix that reflects the overall software structure and highlights dependency patterns between software components. Our work includes the aspect of dependency, however, we focus on a broader set of information in our analysis (e.g., QoS information, usage data).

3 The Anatomy of Web Service Changes

In this section, we analyze changes that are related to Web services. As indicated by the working example of the introduction, changes of Web service happen due to various reasons. During our observations, we've developed a methodology that classifies the factors of influence associated with Web services (see Table 1). We base our classification of the factors of influence on the work of Canfora and Penta [20] that identified the different stakeholders of Web services. Our proposed methodology consists of the following steps:

1. Identify the stakeholders that are interested in the Web service. This is done by analyzing the development process of Web services and investigating how the communication process is structured. For instance, developers might be informed about changes of Web service requirements by phone or by email.
2. Identify the tasks of the corresponding stakeholders. This step is basically the assignment of responsibilities to the interested parties. For instance, a user of a Web service is responsible for giving feedback about the service performance.
3. Collect data of Web services and identify the source of the data. In this step, we distinguish between runtime and static data. Runtime information (e.g., QoS, Web

Table 1. Perspectives on Web services

Perspective	Task
Provider	The provider is responsible for the planing and conception of the Web service. This includes the definition of requirements, the negotiation of SLA with customers, service pricing policy, etc. as well as managing changes of these.
Developer	The main task of the developer is the implementation of the Web service. The developer needs to manage changes of interface descriptions, different versions of the implementation and track change of requirements.
Service Integrator	The service integrator's task is to integrate external services into a software system and focuses on technical aspects of services. Service integrators role is complementary to Web service developers. Service integrators have interest in changes of the interface, QoS attributes and the semantics of the Web service since these have effects on their integration effort of the Web service. Service integrators also modify the requirements of Web services, due to technical issues, like using a .NET client with a Java Web service, etc.
User	The end user of a Web service is the actual consumer of a Web service and has interest in the functionality of Web services from a non technical perspective. The end user specifies the functional requirements and defines QoS attributes that must be satisfied by a Web service.
Broker	The Web service broker manages information of different Web services in a repository that is publicly available for searching and browsing. Web service broker provide information that is needed by Web service requesters to find a particular Web service.

service usage statistics) is provided by tools that continuously monitor Web services. Other Web service related information is (e.g., requirements, user feedback) is entered into a persistence framework by developers, etc.

In our previous work [6], we've analyzed information sources concerning changes of Web services and introduced a hierarchical information model to persist this kind of data as Atom feeds (see Figure 1)[3]. With this information we able to perform analysis on available information and to infer dependencies between changes of different Web service attributes. This empirical data if further classified among different stakeholder perspectives as shown in Figure 2. During the evolution process of a Web service, we can observe transitive effects of changes that lead to a new versions of a Web service (see Figure 3). We consider this change propagation as foundation for the understanding of evolutionary changes of Web services. In this respect, we regard a set of interrelated modifications as step in the evolutionary process of Web services. To illustrate this, let us consider the following illustrating scenario: The provider of a commercial Web service monitors its service continuously in order to obtain usage statistics of Web service. Critical changes in the usage pattern, like a drop in the daily use of a commercial service are very important for the provider. In such cases these changes trigger activities of the Web service provider. For instance, the provider might contact the customer and

[3] See http://berlin.vitalab.tuwien.ac.at/projects/semf for detailed information about the Web service information model.

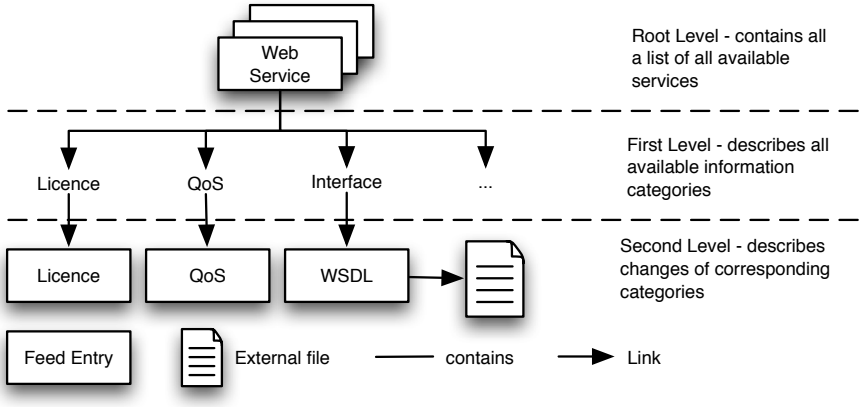


Fig. 1. Web service information model

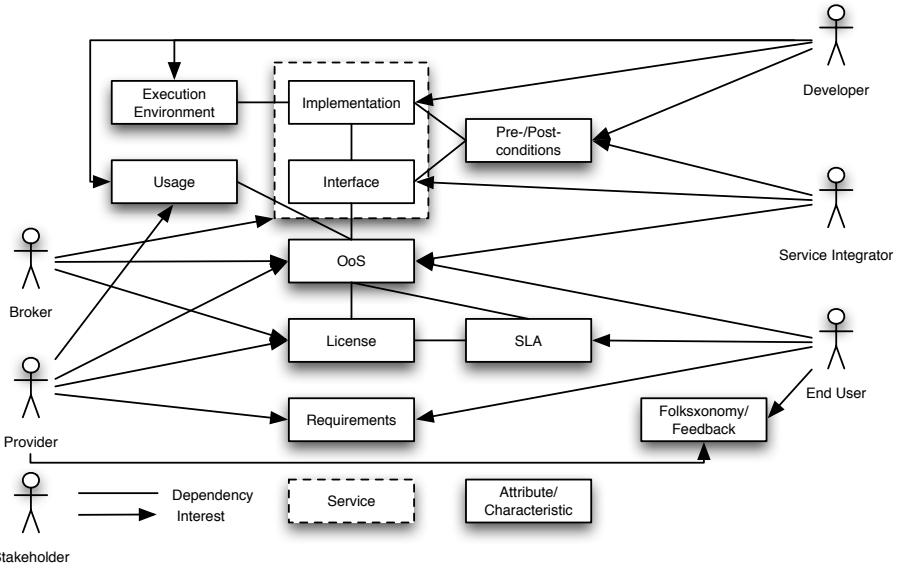


Fig. 2. Information model of Web service changes. Note that the dependencies between the different attributes are not mandatory and depend on the concrete service.

ask for feedback. Let's assume that the customer is not satisfied with the pricing of the service with regard to its performance which is the reason for a change in the service usage. The customer might argue that competitors are able to provide a similar service with a lower price. Since the provider does not want to lose the customer, the provider adapts the pricing and updates the corresponding SLA. Meanwhile the developer was informed by the provider that a customer is not satisfied and that the SLA have changed. The provider requests to optimize the service performance because the provider expects

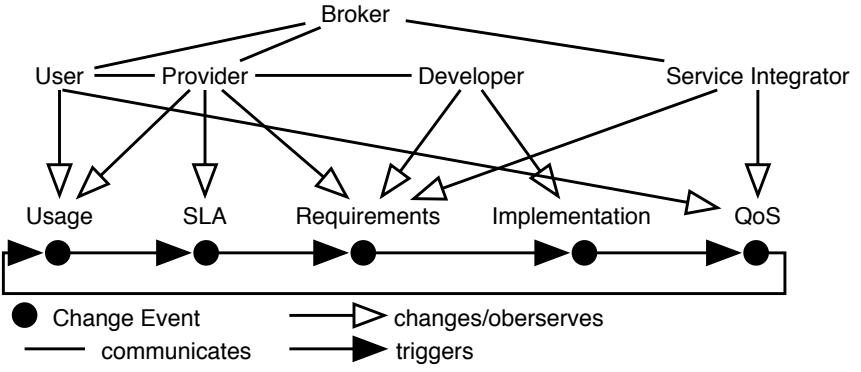


Fig. 3. Propagation of changes and interested parties

Table 2. Impacts on Web service changes

Observed Change	Trigger	Impact on	Modification of	Effect on
Interface	Provider, User, Service Integrator	Integrator, Developer, Broker	Impl.	QoS, SLA, Usage
Implementation	Developer	Integrator, User	Impl.	QoS, Interface
QoS	Usage	Provider, Developer	Impl., Interface	Interface, QoS, SLA, Usage
Usage	User	Provider	SLA	QoS
Requirement	User, Integrator	Provider, Developer	Interface, Impl., SLA	Usage
SLA	Provider	User, Developer, Integrator	Usage	Requirement, QoS, Impl.
Pre-Post Conditions	Provider, Developer	User, Integrator	SLA	Impl.
Feedback	User, Integrator	Provider, Developer	SLA	Usage

in the future customers to require better performance of the service. These changes have effect on the QoS parameters of the Web service which in turn influence the usage of the Web service. Notice that in such scenarios Web service broker are not directly involved. However, after the modifications of a Web service took place, Web service broker might notice changes in the QoS of the service with some delay using QoS monitoring tools and inform potential customers of these adaptations. As shown by the example above, we can observe change propagations and impacts on different perspectives from one single change event. We've summarized potential changes and impacts in Table 2.

In the following subsections we discuss major changes of Web services in detail. We show how change activities are interrelated and explain how these activities contribute to the evolution of Web services. We provide examples that are represented in our

service evolution and discuss the benefits of out approach for the different stakeholders of Web services. Notice that we group changes into two groups. We consider dynamic changes that occur during runtime. These changes can be observed by monitoring tools that log performance related QoS attributes (e.g., response time, throughput, etc.) and collect data of the use of a Web service. Static changes happen prior to the execution of a Web service and may be triggered by observations of the run time behavior of a Web service. The trigger for static changes may be either changes in the observed behavior (decreasing response time due to more server load) or changes in the requirements such as requests for new functionality, etc.

3.1 Web Service Requirements Changes

Changes of requirements are the main driver for all evolutionary Web service changes. Requirements serve as ”benchmark” for the correct functionality of Web service implementations. Changes of requirements are thus very critical during the evolution of Web services and have a number of effects on Web service characteristics:

Implementation: Changes of requirements affect the implementation of Web services, since changes of Web service functionality need to be implemented by the developer.

Interface: The interface reflects changes of Web service requirements when these changes affect functionality of the Web service.

SLA: The provider of the Web service may change SLAs with customers when their requirements change. For example, new functionality needs to be specified in SLAs as well as changes of the required performance of the Web service, data quality, costs, etc.

Pre- and Postconditions: With changing requirements the prerequisites for the execution of Web services might change. For example, new requirements may require a registration for customers prior to the use of the Web service. The effects of the execution of Web service may also change with new requirements. For example, a data Web service might provide additional information to the customer.

All stakeholders of Web services have interest in Web service requirements. Conceptually, requirements can be considered as logical link that links different aspects of Web service modifications together. During the evolution of Web services, every evolutionary step is delimited by the definition of requirements and the publication of a new Web service. Activities by stakeholders as modification of the interface (developer), implementation (developer), definition of SLAs (provider, user), feedback (service integrator, user) are triggered by the definition of requirements, changes respectively.

Example. As noted before, a change of a requirement triggers a set of related activities in order to implement the requirement. Utilizing our evolution framework we can track these activities and link them. When put into a historical perspective, provider can analyze, based on historical information, the costs of the different Web service versions when following the provided links to details about the implementation (see code snippet in Listing 1).

```
<change type="Requirement">
  <link>http://webservice.wisur.at/WISIRISFuzzySearchService?ReqPhon.pdf</link>
  <category>New Function</category>
  <description>A new function was added to the requirement</description>
  <cause type="Feedback">
    <reason>Customers want to search with phonetic methods.</reason>
    <trigger type="User">...</trigger>
  </cause>
<dependencylist>
  <dependency type="Implementation"><!— link to impl. change description —>
  <link id="urn:uuid:e2e2f679-8a67-439a-a65e-bbafd1dd0091"/>
  </dependency>
</dependencylist>
<impact><perspective type="Developer"/>...</impact>
</change>
```

Listing 1. WISIRISFuzzySearch requirement change

3.2 Interface Changes

Syntactical descriptions of Web services define available operations and the structure of messages that Web services are capable to process. We consider interface changes as (i) the addition of new functionality or (ii) the update of existing functionality (e.g., interface refinement with new parameters or removal of functionality). As shown in the overview table, the trigger such changes can be either the provider, the service integrator or the user of a Web service. The cause of an interface change is a change of the requirement. Consequently, interface changes affect primarily the service integrator that is using the service in other software systems, since s/he must adapt the software system that uses the service accordingly. We can observe the following effects of interface changes:

Implementation: Interface changes have impact on the implementation of a Web service. Depending of on the type of interface change, we can observe different effects on the implementation of a Web service. The addition of new functionality or the update of existing functionality are reflected by modifications of the implementation.

QoS: QoS attributes of Web service may be effected by interface changes. However, an interface change does not have immediate consequences for QoS properties. For instance, if the addition of new service functionality also changes existing implementation (e.g., optimizations) then the QoS attributes (e.g., response time, etc.) also change.

Pre- and Postconditions: Changes of the interface have effects on pre- and postconditions of Web services. These reflect the necessary conditions that must be fulfilled to execute a service. For instance, the update of existing functionality to a Web service might require new constraints to be satisfied, such as the provision of a customer identifier and a trial-key.

Usage: Changes of the interface influence the usage of a Web service. Unless the interface change is backward compatible, a new interface with new functionality has impact on the usage of the service.

During the evolution of Web services, each publicly available interface version denotes a new version of the Web service. By analyzing the frequency of interface changes, Web service stakeholders are able to establish the interface stability of the Web service.

Example. From the perspective of the developer it is important to connect interface changes with requirements. In this way it is possible to check implementations for their consistency with (functional) requirements. By combining versioning information with interface modifications, developers are able to track different service versions and corresponding requirements.

From the perspective of the service integrator interface changes are very critical since the service integrator relies on stable interfaces when integrating external services. When interfaces change, service integrators require Web services to be compatible with existing systems. If this is not the case, service integrators require information about the nature of the interface changes to infer how much they have to change. In our approach we follow the classification schema by Leitner et. al.[21] to classify interface related changes.

The example in Listing 2 illustrates how we integrate the changes into our service information model and how we link changes dependencies. In the code snippet, we show how we represent the addition of new functionality to a Web service.

```
<change type="Interface">
  <category type="Add Method"/>
  <description>A new search method was added to the
  WISIRS Fuzzy Search Service</description>
  <cause type="Requirement">
  <— link to the requirement where details can be found —>
   <link id="urn:uuid:823157f7-7174-4b09-b815-64750b0e2f83"/>
  </cause>
<dependencylist>
  <dependency type="Implementation">
<!— link to implementation information in SEMF —>
<link id="urn:uuid:912ae1a0-96b0-11dd-ad8b-0800200c9a66"/>
  </dependency>
</dependencylist>
<impact>
  <perspective type="System Integrator"/>
</impact>
</change>
```

Listing 2. WISIRISFuzzySearch interface change

3.3 Web Service Implementation Changes

Closely related to interface changes are implementation changes. We consider two types of changes, (i) code refactoring/internal optimizations and (ii) change in the functionality. The former subcategory are changes that are transparent for all users of the Web service. The latter is a consequence of interface changes since the new service functionality is expressed by interface changes. Potential triggers for implementation changes are changes of requirements which are caused by the service provider, the user or the service integrator.Implementation changes have effects on the following attributes of Web services:

Interface: Depending on the type of implementation change, we can observe different effects of implementation changes on the service interface. The interface of a service changes when new functionality is added to the Web service or removed from the Web service. Code refactorings or internal code optimizations leave the interface of a Web service untouched since the functionality remains unmodified.

QoS: Internal optimizations have effects on QoS. Consider for instance a database that is accessed by multiple parallel threads simultaneously instead of a sequential manner. This optimization changes the service execution time and is reflected by changes of the response time of the service. Similar, the addition of security mechanisms (e.g. WS-Security, etc.) to a service have impact on QoS attributes.

Pre- and Postconditions: Both, pre- and postconditions are potentially affected by implementation changes. Depending on the type of implementation change, new service functionality (e.g., new methods to search in the provider database) obviously requires new pre conditions (e.g., new input parameters) that must be satisfied in order to execute a service. Postcondition changes depend on the type of implementation change. Consider for example a service that required payment and is now free of charge for customers. In this case, the service implementation was changed in order to acknowledge the new form of service use.

Usage: The effects on the usage of Web services depend on the type of implementation change. Similar as in the case with interface changes, new functionality can lead to an increased usage of a Web service, because potentially more users can be served. Internal changes (along with enhanced performance) might also result in a higher usage of the Web service.

In our evolutionary approach, every evolutionary step is preceded be a series of implementation changes to achieve the fulfillment of requirements. The publication of a new Web service version denotes the finalization of all necessary implementation changes.

Example. As in the case of interface changes, the developer needs to trace modifications in the source code with respect to changes of requirements and user/service integrator feedback. In particular, when the developer modifies the implementation to improve the performance the developer needs to know whether the changes have the desired impact and requires feedback from the user/service integrator. From the perspective of the Web service provider it is important to know how much time was spent to implement in the required modifications. To illustrate these types of change, consider for example a Web service that offers facilities to search in a consumer database. In the example, internal changes were implemented that had no effect on the interface of the Web service. The code snippet in Listing 3 illustrates how we capture these information in our Web service information model.

3.4 Web Service QoS Changes

QoS related changes of Web service depend on changes of other properties of Web services. In contrast to implementation modifications, QoS changes are observed at run time. The reasons for QoS changes are manifold: server load, number of concurrent users, performance of back end systems such as databases, external factors such as network latency, network throughput, as well as issues like security, etc., influence the QoS

```
<change type="Implementation">
<category type="Internal Modification"/>
 <description>The ordering of the search result was changed.</description>
  <cause type="Feedback">
    <reason>User require a ordered search result (by familyname).
   </reason>
    <trigger type="User">... </trigger>
  <link>webservice.wisur.at/</link>
 </cause>
<dependencylist> <— implementing class —>
 <dependency type="Class">
 <name>WISIRISSearchWrapper</name>
<description>Modification of SQL query</description>
  </dependency>
</dependencylist>
<version number="1"> <!— versioning information —>
  <effort developerid="12"> <!— implementation effort —>
  <hours>3</hours>
  </effort>
</version>
<impact><perspective type="Developer"/></impact>
</change>
```

Listing 3. WISIRISFuzzySearch implementation change

attributes of Web services. Domain related QoS attributes, like data quality (completeness, correctness, etc.) when providing data centric services are also of concern. For instance, the hit rate of a search Web service is of importance when a provider desires to sell business reports. Simply put: the higher the hit rate, the higher is the probability that a user will use the service.We can observe the following effects QoS changes:

Usage: Changes of Web service related QoS have impact on the usage of Web services. When a service is selected by QoS attributes like response time, then a degradation of QoS changes such as a higher response time can lead to a reduced service usage.

Implementation: Observed QoS changes may lead to implementation changes. Internal optimizations of the program code (e.g., different algorithms) are potentially used to enhance performance related QoS attributes.

During the evolution of Web services, QoS attributes serve as indicator concerning the overall *fitness* of the Web service. With QoS information, we are to measure the fitness of Web services with regard to SLAs. When put into a historical context, QoS data provides information about the overall development of a Web service and allows to estimate when the performance of a Web service may become critical.

Example. With regard to the provision of data centric services, we address (i) data quality (is the provided information up to date? and (ii) typical QoS (response time, availability, etc.) of a Web service. We now show an example that highlights service quality aspects from the perspective of the service provider with regard to service performance. The code snippet in Listing 4 shows a notification about the violation of SLA constraints that is generated by a monitoring tool that logs the performance of Wisur's Web services, making the observation of QoS very important from the perspective of the service provider.

```
<change type="QoS">
  <category type="Violation"/>
  <cause type="Usage">
    <reason>Response of WISIRISFuzzySearchService</reason>
    <trigger type="Service Environment">...</trigger>
  </cause>
<dependencylist>
<dependency type="SLA"><!— link to sla information in SEMF —>
  <link id="urn:uuid:da66f3c0-96da-11dd-ad8b-0800200c9a66"/>
 </dependency>
</dependencylist>
<impact>
  <perspective type="Developer"/><perspective type="Provider"/>
</impact>
<details>
  <classes><class name="WISIRISDataAccess">
  <executiontime>59955ms</executiontime>
  <exception/>
  </class></classes>
</details>
</change>
```

Listing 4. WISIRISFuzzySearch QoS change notification

Notice that our tool includes information for the developer in order to track the part of the Web service implementation which is responsible for the violation of the SLA. Similar to Web service providers, end users are concerned with the Web service quality. Consider the example, of a Web service which must respond within 60 seconds and be available 24/7. The data presented in Figure 4 and Figure 5 shows the observed execution times of the reporting service of two consecutive months of a real world Web service from Wisur. As shown in the figure, the execution time during April 2008 was constantly under 30 seconds, with a tendency to increase towards the end of the month and a constraint violation at the end of April 2008. This lead to user feedback and triggered a change in the implementation of the Web service. The observed execution time in May 2008 was considerably higher (more peaks moving towards 60 seconds) but no constraint violation occurred.

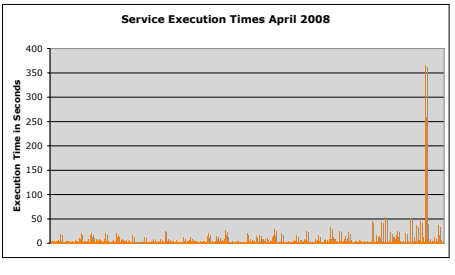


Fig. 4. Observed execution during April 2008

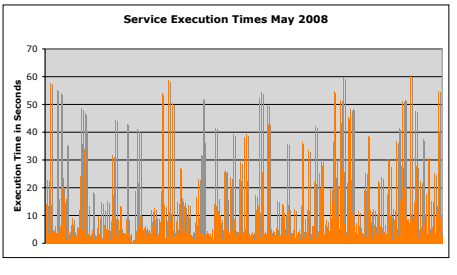


Fig. 5. Observed execution during May 2008

4 Discussion and Outlook

In this paper, we analyzed dependencies of Web service changes and provided a model that captures the changes. We introduced a conceptual model which offers the means for deeper analysis of these changes. In context of Web service evolution we are able to define an *evolutionary step* as set of activities (modifications of the interface, implementation, requirements, SLA) that are triggered by different stakeholders of Web services. The result of an evolutionary step is a new version of a Web service that is adapted to these changes.

This lays the foundation for the Web service evolution process. We consider Web service evolution as an (potentially) *indefinite sequence of evolutionary* steps that result in observable changes of the Web service. We assume that there are several *variations* of a Web service at a given point in time. Every variation represents a independent evolution sequence of a Web service and is represented by historical information.

In future work, we will focus on composite Web services and investigate evolutionary issues of Web service compositions and investigate graphical models for the representation of the evolution [22] of complex composite Web services. Moreover, we are going to formalize our proposed conceptual methodology with a meta model that provides a formal foundation for roles and change dependencies. Furthermore, we will investigate complex event processing with regard to evolutionary aspects. In this context, we plan to extend our framework with the support of event processing in the context of Web service registries as discussed in [23].

References

1. Barros, A.P., Dumas, M.: The rise of web service ecosystems. IT Professional 8, 31–37 (2006)
2. Chinnici, R., Moreau, J.J., Ryman, A., Weerawarana, S.: Web Services Description Language (WSDL) 2.0 (2007)
3. W3C: OWL Web Ontology Language Overview (2004) W3C Recommendation (February 10, 2004)
4. Dumitru, R., de Bruijn, J., Mocan, A., Lausen, H., Domingue, J., Bussler, C., Fensel, D.: Www: Wsmo, wsml, and wsmx in a nutshell. In: Mizoguchi, R., Shi, Z.-Z., Giunchiglia, F. (eds.) ASWC 2006. LNCS, vol. 4185, pp. 516–522. Springer, Heidelberg (2006)
5. Akkiraju, R., Farrell, J., Miller, J., Nagarajan, M., Schmidt, M.T., Sheth, A., Verma, K.: Web Services Semantics – WSDL-S (2005)
6. Treiber, M., Truong, H.L., Dustdar, S.: Semf - service evolution management framework. In: SEAA 2008 (to appear, 2008)
7. OASIS: Web Services Distributed Management: Management of Web Services (WSDM-MOWS) 1.1 (2006)
8. Catania, N., Kumar, P., Murray, B., Pourhedari, H., Vambenepe, W., Wurster, K.: Web services management framework, version 2.0 (2003)
9. Casati, F., Shan, E., Dayal, U., Shan, M.C.: Business-oriented management of web services. Commun. ACM 46, 55–60 (2003)
10. Fang, R., Lam, L., Fong, L., Frank, D., Vignola, C., Chen, Y., Du, N.: A version-aware approach for web service directory. In: ICWS, pp. 406–413 (2007)

11. Kaminski, P., Müller, H., Litoiu, M.: A design for adaptive web service evolution. In: SEAMS 2006: Proceedings of the 2006 international workshop on Self-adaptation and self-managing systems, pp. 86–92. ACM, New York (2006)
12. Ghezzi, C., Guinea, S.: Run-time monitoring in service-oriented architectures. In: Test and Analysis of Web Services, pp. 237–264. Springer, Heidelberg (2007)
13. Conradi, R., Westfechtel, B.: Version models for software configuration management. ACM Comput. Surv. 30, 232–282 (1998)
14. Lehman, M.M., Ramil, J.F.: Software evolution: background, theory, practice. Inf. Process. Lett. 88, 33–44 (2003)
15. Cook, S., Harrison, R., Lehman, M.M., Wernick, P.: Evolution in software systems: foundations of the spe classification scheme: Research articles. J. Softw. Maint. Evol. 18, 1–35 (2006)
16. Lehman, M.M.: Laws of software evolution revisited. In: Montangero, C. (ed.) EWSPT 1996. LNCS, vol. 1149, pp. 108–124. Springer, Heidelberg (1996)
17. Papazoglou, M.: The challenges of service evolution. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 1–15. Springer, Heidelberg (2008)
18. Andrikopoulos, V., Benbernou, S., Papazoglou, M.: Managing the evolution of service specifications. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 359–374. Springer, Heidelberg (2008)
19. Sangal, N., Jordan, E., Sinha, V., Jackson, D.: Using dependency models to manage complex software architecture. SIGPLAN Not. 40, 167–176 (2005)
20. Canfora, G., Di Penta, M.: Testing services and service-centric systems: Challenges and opportunities. IT Professional 8, 10–17 (2006)
21. Leitner, P., Michlmayr, A., Rosenberg, F., Dustdar, S.: End-to-end versioning support for web services. In: IEEE International Conference on Services Computing, SCC 2008, vol. 1, pp. 59–66 (2008)
22. Luqi: A graph model for software evolution. IEEE Transactions on Software Engineering 16, 917–927 (1990)
23. Michlmayr, A., Rosenberg, F., Leitner, P., Dustdar, S.: Advanced event processing and notifications in service runtime environments. In: DEBS, pp. 115–125 (2008)

Standardization as a Business Ecosystem Enabler

Paul L. Bannerman and Liming Zhu

Managing Complexity, NICTA, Australian Technology Park, Sydney, NSW, Australia
School of Computer Science and Engineering, University of New South Wales, Australia
{Paul.Bannerman,Liming.Zhu}@nicta.com.au

Abstract. This paper reports research-in-progress that considers standardization as an enabler in business ecosystems. Based on issues encountered, we propose that: business model design is also critical for standards bodies; standards can separate competition from cooperation; standards employ rule-centric designs; and requirements specification and compliance checking are essential. The propositions are illustrated using a standards body with which we have been working.

Keywords: Standardization, business model, service-oriented computing.

1 Introduction

As service-oriented computing enables businesses to expose their functions more widely, a business ecosystem forms around the exposed services. To foster efficient collaboration between the stakeholders (many of whom are competitors), a standardization body is often formed to improve interoperability within the ecosystem through standards. This is particularly the case in vertical industries. Examples include HL7 in health, ACORD in insurance and MISMO in the US mortgage industry. This type of standard can be difficult to develop and implement because an existing industry structure is often entrenched, and a new e-business standard may challenge that structure. Change may impact existing business models requiring them to be rebalanced to establish a new equilibrium. It is important, therefore, to consider how such bodies and standards can be enablers of the business ecosystem.

NICTA (nicta.com.au) has been working with a leading Australian e-business industry standardization body, LIXI (Lending Industry XML Initiative; lixi.org.au), which serves the lending industry in Australia. LIXI initially developed a XML-based data-centric standard, later complemented by a process model described in BPMN (Business Process Modeling Notation) developed by NICTA. To further bridge the gap between business standards and technical implementation, and promote technical interoperability, NICTA also helped devise a Reference Architecture and associated Reference Implementations to supplement LIXI's e-business standards [9, 10]. During this process, we encountered a number of business and technical issues that can impact the effectiveness of standardization in a business ecosystem:

1) *Process openness and funding model*. Striking the right balance in the role played by the standards organization in the ecosystem involves trade-offs between

G. Feuerlicht and W. Lamersdorf (Eds.): ICSOC 2008, LNCS 5472, pp. 298–303, 2009.

being open to as many industry players as possible (large and small) and sourcing adequate funding to finance operations.

2) *Intellectual Property Rights (IPR) and commercial liability*. Policies relating to in-bound and out-bound IPR can significantly impact members' participation in standardization processes, their ability and willingness to adopt developed standards and the commercial risk they may face if they implement the standards.

3) *Consistency and variety balance*. There are tradeoffs between prescriptive guidance and flexibility, immediate technical needs and long term evolution. This balance has a major impact on the diversity and efficiency of the business models in the business ecosystem.

4) *Interpretation of standards*. Multiple interpretations of a standard can result in reduced interoperability between systems. The risk is that an application will have to be changed significantly whenever it needs to interact with a new system, even though both parties could reasonably claim to be in compliance with the e-business standards. Such hindrance to genuine mass interoperability can reduce the efficiency of the whole business ecosystem.

In this short paper, we analyze these issues and outline propositions that relate to the business of a standards body (addressing issue 1 and 2) and the nature of its standards (addressing issue 3 and 4) as enablers in a business ecosystem of diverse interests and business models. We develop the propositions in Section 2 and illustrate them against the LIXI case in Section 3 before making concluding comments.

2 Issue Analysis

Existing research is very limited in addressing these issues. For issues 1 and 2, while some work has been done in open communities and IPR (e.g., [3]), theory remains under-developed. Standards organizations adopt different strategies, but the principles underlying decision options and criteria are unclear. Yet it is clear that these decisions can impact the efficiency and performance of stakeholders and the ecosystem at large. For issue 3, our early work has shown that a rule-based rather than structure-based architecture approach may alleviate the problem of balancing flexibility for future evolution against prescriptive guidance to meet short-term technical needs [11]. However, the connection to business models is not fully considered. For issue 4, due to the nature of standards, the solutions to standard compliance checking vary widely. At one extreme, an Application Programming Interface (API) standard may have an automated and comprehensive compliance testing tool kit. At the other, a process standard may rely on subjective assessment through checklists, interviews and documentation. Data format standards present a particular problem. Superficially, "schema validation" is a good solution. However, schemas cannot usually capture all the rules and constraints between data elements that must be satisfied. We advocate non-prescriptive requirements specification and strong compliance checking.

2.1 The Business of a Standards Organization

Standards organizations typically operate within a business ecosystem. A business ecosystem is an interacting population of business entities in a bounded environment.

Each has its own agenda but must interact with others to survive and prosper. In the ecosystem, entities co-evolve capabilities that enable them to work cooperatively and competitively in pursuing their own objectives [7]. Standards can be an important capability in this ecology because they can enable the necessary interactions.

A business model is an organization's fundamental design for creating value. It comprises a business's value proposition, target market(s), how it will generate revenue, its cost structure, and activity value chain [4]. An e-business model, for example, is optimised around transacting business through electronic media such as the web. In conjunction with capabilities, these components permit great diversity in business models. There is no single 'right' model for an industry. Indeed, competition drives differentiation in the business models of industry participants. However, for optimum performance, business model design must be consistent with the aims and purpose (the 'dominant logic') of the entity. The parts must fit together harmoniously, otherwise inefficient integrating mechanisms will evolve to overcome the misfit [6].

We propose that *business model design is also critical for standards bodies and their performance*. The challenge for a standards body in formulating its appropriate business model within a business ecosystem can be greater than that of the business operatives. By definition, a standards organization serves a diverse range of industry stakeholder interests. Striking the right balance between those interests – that is, establishing its dominant operating logic – may be a process of co-evolution with its members. Key decisions include: Should it engage in commercial for-profit activities or restrict itself to providing not-for-profit services to members? How will it fund its operations; at the front-end from membership fees or at the back-end from standards licensing and publication fees? How open should it be in inviting input to standards development and access to completed standards [5]? Should it tightly control IPR to published standards or push it out to the industry, and how should it approach IP from others [8]? Finding the right balance in answering these questions will fundamentally shape the business model and internal consistency of the standards organization.

Furthermore, we argue that in a business ecosystem *standards can sequester competition from cooperation*, enabling companies to coexist and thrive. Resource-based theory [2] tells us that organisations comprise a variety of resources including: ordinary capabilities that are necessary for daily operations (and which may be common to other firms); and unique capabilities that can generate distinctive value for the firm. These latter capabilities enable innovation [1] and competitive advantage [2]. For firms that need to interoperate within an ecosystem, a standard can be a common operational capability that permits firms to interact, as necessary, through their business processes, in a non-threatening manner. This sequesters interoperation from the exploitation of strategic capabilities and the competitive side of the industry.

2.2 Nature of a Non-prescriptive But Compliance-Checkable Standard

Our experiences working with the Australian lending industry show that the cost of pair-wise integration can be prohibitively high even when both sides claim to be data standard "compliant". From a service-oriented computing perspective, problems in integrating multiple parties are not new. Some degree of control, centralized coordination and using SOA/WS-* standards can alleviate many of them. However, these problems are significantly different in the context of a whole industry, which is

essentially an ultra-large-scale business ecosystem. Too much prescription (especially beyond data format) in industry-wide standards may dictate particular business model designs or behavior and constrain innovation. The challenge is to determine what to standardize, beyond data and leveraging service-oriented computing, so that a right balance between too much and not enough prescription can be reached. Accordingly, we advocate two further propositions:

Use rule-centric architectures. In Service Oriented Architecture (SOA), the notion of architecture traditionally implies an arrangement of structural components and connectors. An industry-level SOA cannot prescribe too much structure as it may prevent new business relationships/models between parties and adoption of new technologies. We advocate a rule-based approach. An architectural rule is defined as principles that need to be followed by structures within a system. An architectural rule may serve several potential structural architectures. This approach is consistent with other IT-related industry practices such as open API and mashup (e.g., Google's OpenSocial). We illustrate this rule-based approach in Section 3.

Support standard requirements specification and compliance checking. Often, there are two difficulties in standard compliance checking. First, standards bodies do not provide a consistent way of checking standard compliance. As argued earlier, mechanisms like schema validation are inadequate. Second, compliance checking may fail because a standard does not sufficiently capture the requirements. The implementer faces a trade-off between losing must-have features and not complying with a standard. These difficulties are usually caused by a weak standard requirements elicitation phase without a compliance checking tool kit. Thus, we propose that:

1. Standard requirements should also be subject to an open process with voting as a form of quality control. Currently, most standardization bodies only consider the standard itself to be subject to the standardization process.
2. All standard specifications should be "testable" for compliance checking. Non-testable specifications will cause downstream interoperability problems.
3. A compliance testing toolkit must be released with the final draft of the standard before voting and after standard publication to facilitate interpretation of the standard.
4. A reference implementation that has passed the compliance testing toolkit should also be released.

Items 3 and 4 have been adopted in a small number of standards bodies such as Java Community Process (JCP). However, the JCP standards are all Java APIs which are easily testable. The challenge is to find a suitable way of testing other types of standards. For items 1 and 2, many standards bodies do vote on a new standard proposal. However, the proposal usually only covers high level purposes rather than detailed standard requirements, let alone testable requirements in conjunction with a compliance testing toolkit.

3 Case Illustrations

Application of these propositions is illustrated from the LIXI case.

3.1 The Business of a Standards Organization

In an effort to encourage broader industry participation, especially from small and medium enterprises, LIXI recently changed its business model. Previously, standards were free to members and funding was sourced from memberships. In the change, membership fees were reduced to a nominal annual figure and a schedule of licensing fees was introduced for different bundles of standards. This increased the openness of the standard development process in terms of broader participation.

However, two challenges emerged: management of IPR, and; commercial liability (e.g., how can LIXI avoid legal liability if a standard beaches another organization's IPR, and how can it avoid being sued by members if their business suffers as a result of using a LIXI standard?). Elements of these challenges are purely legal, for which exemplars exist in other successful standards organizations. Others affect the business models within the ecosystem and the interplay between openness, funding, IPR control, cooperation and competition. Based on the above theory, these challenges need to be considered in the context of the organization's purpose and business model. An open design that enables interoperation between members in a manner that is not perceived to be commercially threatening could achieve a consistent logic that fosters effective cooperation.

3.2 Nature of a Non-prescriptive But Compliance-Checkable Standard

The following are example rules from a list of 40 applied in the LIXI context [10]:

Use Minimal Service Interface. A LIXI-compliant system should use message-centric (rather than operation-centric) interfaces. That is, service interfaces should not expose abstractions in the form of remote procedures. Essentially, we advocate the use of a single operation on a service (ProcessLIXIMessage), but allow more complicated interfaces to exist. Messaging behaviors are specified by LIXI content structure and LIXI message exchange protocols. This rule encourages maximum flexibility in the face of constant evolution. Ever-changing shared contexts are carried within LIXI messages. Message processing logic can either be hidden behind the service or exposed as protocol-related metadata.

Avoid Explicit Intermediaries. We do not introduce the role of an intermediary explicitly in the reference architecture. However, we allow such intermediaries to organically appear in the overall ecosystem. This is very different from existing e-business meta-standards such as ebXML, which have an explicit concept of central registry and repositories through which companies post business processes, capability profiles and collaboration protocol agreements. Technically, this is appealing and simplifies some business scenarios. However, we found it very difficult to introduce such a structure within LIXI because of complex business issues such as who the intermediaries should be, legal issues such as confidentiality concerns, and practical issues such as the difficulty of semi-automated agreement negotiation. Thus, in our reference architecture, interacting directly with another business or indirectly through an intermediary is treated as the same general mechanism. Local intermediaries within certain areas can be introduced. Dynamic binding and proxy solutions can help achieve various relationships in practice.

Regarding testing compliance, we provided a number of reference implementations with data checking beyond the basic schema validation [9, 10]. The use of RELAX-NG and Schematron are proposed for better compliance checking.

Finally, we are currently developing a more systematic compliance testing toolkit strategy and standard requirements elicitation/voting process.

4 Conclusions

This paper has considered the role of standards and standardization bodies as enablers in business ecosystems and developed initial business and technical propositions with practical application. In addition to the basic function of enabling interoperation between industry stakeholders, we have proposed that: the effectiveness of industry participants, including standards bodies, is a function of the internal consistency of their business model; standards are useful in demarking a boundary between cooperation and competition; effective standards use rule-based architectures, and; standardization must include requirements specification and compliance checking.

NICTA continues to work with LIXI to research and apply these principles.

Acknowledgments. NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

1. Bannerman, P.: Smoothing Innovation Discontinuities. In: Proceedings of the IEEE International Conference on Communications (ICC 2008), Beijing, pp. 5458–5462 (2008)
2. Barney, J., Clark, D.: Resource-based Theory: Creating and Sustaining Competitive Advantage. Oxford University Press, Oxford (2007)
3. Blind, K., Thumm, N.: Intellectual Property Protection and Standardization. International Journal of IT Standards & Standardization Research 2(2), 61–75 (2004)
4. Chesbrough, H., Rosenbloom, R.S.: The Role of the Business Model in Capturing Value from Innovation. Industrial and Corporate Change 11(3), 529–555 (2002)
5. Krechmer, K.: Open Standards Requirements. International Journal of IT Standards and Standardization Research 4(1), 43–81 (2006)
6. Miles, R., Snow, C.: Organizational Strategy, Structure, and Process. Stanford Business Classic (2003)
7. Moore, J.: Predators and Prey: A New Ecology of Competition. Harvard Business Review 71(3), 75–86 (1993)
8. Pisano, G.P., Teece, D.J.: How to Capture Value from Innovation: Shaping Intellectual Property and Industry Architecture. California Management Review 50(1), 278–296 (2007)
9. Zhu, L., Thomas, B.: LIXI Visible Loans: Reference Architecture and Implementation Guide. Lending Industry XML Initiative (LIXI) (2007)
10. Zhu, L., Thomas, B.: LIXI Valuation Reference Architecture and Implementation Guide 1.0. Lending Industry XML Initiative (LIXI) (2007)
11. Zhu, L., Staples, M., Tosic, V.: On Creating Industry-Wide Reference Architectures. In: The 12th IEEE International EDOC Conference (EDOC 2008), Munich, Germany (2008)

Managing Quality of Human-Based eServices

Robert Kern, Christian Zirpins, and Sudhir Agarwal

University of Karlsruhe, Karlsruhe Service Research Institute, Germany
{robert.kern,christian.zirpins,sudhir.agarval}@ksri.uni-karlsruhe.de

Abstract. Modern business strategies consider Web-based outsourcing of microjobs to the masses. Respective business activities are however difficult to manage. Traditional approaches of covering human tasks in business processes build on assumptions of limited process scale and closed organizational models that are not valid in crowdsourcing scenarios. Web services have been proposed as a means to represent human tasks that allow leveraging interaction, brokerage and composition capabilities of SOC for human interaction management. In this paper we argue that crowdsourcing requires considering qualitative constraints and sketch a platform for managing the quality of human-based eServices.

Keywords: Crowdsourcing, Human-based eServices, QoS Management.

1 Introduction

Despite all endeavours of rationalizing their business processes, organizations still face many types of tasks which cannot be fully automated but which require human intelligence or action to be completed. Traditionally, such activities are integrated into workflows as so called *human tasks*, which are passed to a well defined group of employees of the company. The drawback of this approach is its limited scale. A large workforce has to be hired to cover peak workloads that partly idles in times of small workload resulting in cost overheads.

Amazon was first to address this issue by offering the Web marketplace *Mturk* (http://www.mturk.com), on which service requesters can publish open calls for *Human Intelligence Tasks (HITs)* [1]. Any Internet user that meets certain skill criteria might act as a service worker and complete tasks for small amounts of money. The business model of the platform is to act as a broker between requesters and workers and to keep a percentage of the service fee for each task. However, there is no control over the correctness of the results that are delivered by the services workers, nor about any other quality aspects like response time or availability. In case a certain QoS-level is required, the requester has to take care of it himself. This limitation dramatically restricts possible usage scenarios.

In this paper we argue that the utilization of human-based electronic services for large-scale outsourcing of human tasks requires platform support for actively managing their QoS and offer initial considerations on respective platform mechanisms. In the following sections, we will first elaborate on the economical background of human-based electronic services and introducing the term *people*

G. Feuerlicht and W. Lamersdorf (Eds.): ICSOC 2008, LNCS 5472, pp. 304–309, 2009.

service for a broad type of human tasks in sec. 2. Following that, we highlight some requirements for respective service-oriented platforms and infrastructure mechanisms in sec. 3. Finally in sec. 4, we outline our conceptual solution approach for an integrated platform that does not only act as a broker between service requester and service worker but which extends the business model by actively managing the QoS of the results delivered to the service requesters.

2 Human-Based Electronic Services

Within the diversified Web 2.0 technology landscape, a recent trend of social computing applications concerned the virtualization of human-based intellectual activities as electronic services. The business motivation for these developments can be identified as a combination of two orthogonal economic strategies: human capital outsourcing and business process rationalization. Human capital outsourcing offers benefits of increasing labor capacity and human potential in an organization without the indirect costs and risks that accompany long term employment relationships. Business process rationalization aims at optimizing and automating sourcing interdependent activities that in combination, add value to an organization in order to maximize revenue.

While rationalization is a classical domain of information systems, the outsourcing of labor witnesses enormous changes through the advent of modern Web technologies. On an individual basis, connectivity and interoperability of Web-based software applications fostered an increase of *homeshoring* so called *telecommuters* that offer their services to independently carry out a wide array of IT-enabled jobs via outsourcing portals like Odesk (`http://www.Odesk.com/`) or Elance (`http://www.Elance.com`). At the same time, the phenomenon of *wikinomics* opened up new possibilities of mass collaborations that allowed to drastically increase the scale of outsourcing labor while simultaneously decreasing activities into specialized microjobs. Respective *crowdsourcing* generally involves an organizational requester issuing calls to members of the public that carry out activities on individual or group basis [2]. Despite its potentials it has to be said that this approach also involves severe risks. Companies easily underestimate the challenges of managing crowdsourced projects that include issues like coordination, motivation, contracting, sustainability and loyalty. Another dimension of issues revolve around the virtual absence of employment rights.

For companies, the next step is to leverage wikinomics to a business context, where competition requires not only effectiveness but efficiency, and rationalization of economic activities is a necessity. This corresponds to a sharper focus on qualitative properties of functional business activities. For example, development of acceptable Wikipedia entries might take an unconstrained duration and is driven by public volunteers for idealistic reasons. In contrast, the documentation of product features in the crowdsourced development of a product manual needs to be finished in a given time and to a required standard in return of a monetary compensation in order to provide business value for the requester. Theoretical foundations for the qualitative optimization of massively collaborative processes might be found in areas like game theory or cybernetics. Their

application however requires infrastructures for large scale human interaction management in the context of mass collaboration. Such infrastructures are increasingly provided on the basis of electronic services that represent microjobs in crowdsourcing contexts and allow for their systematic and automatic sourcing and coordination. We refer to these as *people services* (pServices) and define them as Web-based software services that enable human intelligence, perception, or action to be effectively sourced and coordinated in order to contribute to massively distributed business processes.

An early example of pService infrastructure is Amazon's Mturk platform that focuses on pService sourcing and provides a simple request-driven marketplace for so called human intelligence tasks (HITs). Mturk gained public attention during its application for searching half a million of satellite images for the lost vessel of Microsoft's Jim Gray to thousands of volunteers on the Web [3]. The failure of this undertaking might be considered an indication for a number of problems and open research questions that still exist for people service infrastructures. While problems like misunderstanding of requests and missing information on the results of others indicate shortcomings of pService sourcing and coordination, the fact that the mission was just not finished in time underlines the need to actively manage the quality of people services.

Building on pServices, organizational requesters might plan and operate massively distributed business processes by mapping their operational business models onto service-oriented architectures and platforms that are enabled for crowdsourcing. Such architectures and platforms leverage basic software service technologies like WS Human Tasks [4] that supports the implementation of human-based Web Services. However, the distinctive nature of crowdsourced business activities requires for rethinking the upper layers of the software service stack. These traditionally focus on workflows of classic business processes in a context of closed organizational models. Situations, where thousands of tasks need to be allocated to participants on the Web are so far not considered.

3 Requirements Analysis

Corresponding to the business goals of enabling crowdsourced business activities with qualitative optimizations, pService platforms have to provide means for two major classes of tasks: sourcing and coordination. pService sourcing leverages software service brokerage and marketplaces in order to discover and aggregate temporary workforce. pService coordination leverages software service mashups and composition in order to plan and enforce collaboration of temporary pService workers. An important aspect for pService platforms is to support service workers to understand pService semantics and provide ergonomic interfaces that incorporate mobile devices [5]. Additionally, a crucial point is for the platform methods to jointly provide a certain level of quality as required by organizational requesters. Only if qualitative properties like performance, scalability, availability, and correctness in relation to prices are observable and guaranteed, organizations will consider to incorporate pServices. Therefore pService sourcing

not only needs to map pService requests to respective offers but also to ensure timely allocation with acceptable costs. Likewise, pService coordination not only needs to regulate and enforce relationships of multiple pService tasks but also to introduce feedback loops for controlling correctness of results. The following list summarizes key quality requirements for pServices.

- *Performance:* Ability to return a result within certain response time limits
- *Scalability:* Ability to handle a certain average and peak number of requests
- *Availability:* Ability to continuously provide the service
- *Correctness:* Ability to return a minimum percentage of correct results

These quality requirements can typically not be met by individual service workers. Neither do these 'scale', nor are they necessarily available at any time requested or at once. Moreover, individual service workers won't be able to deliver correct results for all service requests. Thus, it seems inappropriate for service requesters to directly deal with individual service workers. Instead, their work should be managed by a pService platform that takes care of quality-of-service goals. Subsequently, we differentiate 2 types of pServices: individual ones performed by service workers and the managed ones requested by the service requesters for which SLOs can be defined and which might build on combinations of individual ones. We call the latter *managed people services* and define them as pServices that support service level objectives (SLOs) to guarantee certain quality-of-service goals like performance, scalability, availability, and correctness.

4 Platform Considerations

We see a need for people service platforms to support SLOs and guarantee QoS to service requesters. In the following, we provide some preliminary considerations about the functionality of such a platform. Fig. 1 describes the basic components.

Similar to Mturk, the proposed platform deals with two groups of customers: service requesters, who are submitting pService requests to the platform and service workers, who are willing to work on those requests. However, in our case, the pService requests are not directly passed to the service workers, but the platform deals with two different types of pService requests. *Managed pService requests* which are passed to the platform by the service requesters and *(native) pService requests* which are passed to the service workers by the platform.

A managed pService request consists of two parts: a service template and a set of SLOs. The service template refers to what is presented to service workers when delivering services. It includes informal and possibly semantic descriptions of the problem to be solved as well as a response form to interact with service workers when delivering results. The SLOs comprise information about response time goals, scalability and availability requirements as well as the correctness of the results. Availability goals include the initial availability of the service (date, time), as well as the period of time it will be available for. Service workers can register at the platform and provide information about their skills, availability, and expected compensation with respect to service template classes.

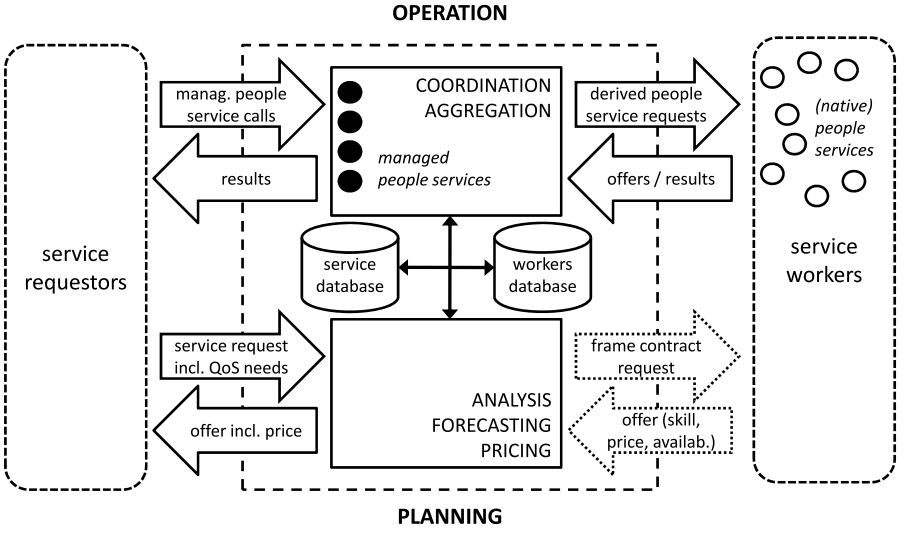


Fig. 1. Schematic view of proposed pServices platform

Once a managed pService request is submitted to the platform, the planning component analyses the request. This includes forecasting whether enough service workers with appropriate skills will be available. In addition, an execution plan is created that addresses the service level objectives. Based on the type of service and the service level objectives, the platform generates a price offer for delivering the service to the requester as a managed pService.

At runtime, the service requester passes individual service calls to the platform. The platform operation engine will autonomously coordinate those service calls based on the execution plan which might make use of a variety of approaches to actively manage the SLOs: An incoming service call might be turned into multiple pService calls passed to service workers and results returned by one service worker might be compared with results from other service workers or sent to other service workers for validation. The quality of the service worker's contributions might be tracked with a reputation mechanism and might be used for quality forecasting. A notification mechanism might be used to actively pass requests to individual service workers to reduce lead time before they start working on a service. The same request might be sent to multiple service workers in parallel to ensure that at least one of them meets the response time goals. Variable incentives might be used to drive the service worker's motivation to meet response time goals, to be available even at unusual hours, and to deliver correct results. These and possibly more options are offered by the platforms as patterns that might be utilized in order to provide certain qualities of pServices. Many of those patterns build on the coordination of interactions between service workers in order to realize redundancy as well as feedback control loops. After completing the execution plan, the individual responses returned from the service workers are aggregated and returned to the service requester as a single result.

The business model of the platform is based on the gap between the fixed price for the managed pServices payed by the service requester and the average incentives payed to the service workers. Based on information about registered service workers and historical information, the future incentives might be estimated and used for pricing of new pServices.

5 Summary and Outlook

There are two general approaches for incorporating human activities into electronic business processes or applications, each having a major deficiency. Traditional human tasks of business processes lack scalability because they are typically performed by a closed group of people. Recent approaches like Amazon's Mturk are addressing the scalability issue by dealing with a virtually unlimited number of service workers over the Internet but at the same time lack quality because of limited control of the workforce.

After introducing the term of people service (pService) for a broad type of electronic services that leverage human intelligence, perception, or action, this paper discussed the need for pServices to meet quality-of-service goals. An integrated platform and business model was outlined, which is capable of supporting SLOs for pServices by actively managing contributions of service workers.

Additional research is required for many aspects of the platform, e.g. the formal description of the service requests which allow for matching them to the skill profiles of service workers, reputation mechanisms and incentive models for motivating the workers to do their best, and coordination patterns to produce high quality results based on the contributions of multiple service workers. We plan to substantiate our concept by implementing a prototype for the proposed pService platform and validate the concept based on different usage scenarios.

References

1. Pontin, J.: Artificial intelligence, with help from the humans. The New York Times (March 25, 2007), http://tinyurl.com/58nec5
2. Brabham, D.C.: Crowdsourcing as a model for problem solving: An introduction and cases. Convergence: The International Journal of Research into New Media Technologies 14(1), 75–90 (2008)
3. Silberman, S.: Inside the high-tech search for a silicon valley legend. Wired magazine (July 24, 2007), http://tinyurl.com/358k7f
4. Agrawal, A., Amend, M., Das, M., et al.: Web Services Human Task (WS-HumanTask), Version 1.0. Technical report, Active Endpoints Inc., Adobe Systems Inc., BEA Systems Inc., IBM Corporation, Oracle Inc., and SAP AG (2007)
5. Maleshkova, M., Komazec, S., Grasic, B., Denaux, R.: iService: Human Computation through Semantic Web Services. In: Future Internet Symposium (FIS 2008) (October 1, 2008), http://tinyurl.com/5fr3po
6. Sycara, K., Paolucci, M., Ankolekar, A., Srinivasan, N.: Automated Discovery, Interaction and Composition of Semantic Web Services. Journal of Web Semantics 1(1), 27–46 (2003)

Third Workshop on Trends in Enterprise Architecture Research (TEAR 2008)

Introduction: Third Workshop on Trends in Enterprise Architecture Research (TEAR 2008)

Stephan Aier[1], Pontus Johnson[2], and Joachim Schelp[1]

[1] Institute of Information Management, University of St. Gallen,
Müller-Friedberg-Strasse 8
9000 St. Gallen, Switzerland
{stephan.aier,joachim.schelp}@unisg.ch
[2] Dpt. of Industrial Information and Control Systems, Royal Institute of Technology (KTH),
Stockholm, Sweden
pontus@ics.kth.se

1 Introduction

The field of enterprise architecture attracted the attention of the research community for the first time when Zachman introduced the Framework for Information Systems Architecture in 1987. However, it was not until 1996 that enterprise architecture emerged as an active field of business activity and research.

Enterprise architecture (EA) is important because organizations need to adapt with increasing speeed to changing customer requirements and business goals. This need influences the entire chain of activities of an enterprise, from business processes to IT support. To keep the enterprise architecture coherent and aligned with business goals, the relations between the different architectures must be clearly defined and consistent.

In previous years, the emergence of service oriented design paradigms (e.g. Service Oriented Architecture, SOA) contributed to the relevance of enterprise architectures. The need to design services along business processes forced companies to pay more attention to business architectures. At the same time, the growing complexity of existing application landscapes lead to increased attention to application architectures.

To better align business and IS architectures a number of major companies started to establish EA efforts after introducing service oriented architectures. Until recently, practitioners, consulting firms and tool vendors have been leading in the development of the EA discipline.

Research on enterprise architecture has been taking place in relatively isolated communities. The main objective of this workshop is to bring these different communities of EA researchers together and to identify future directions for EA research with special focus on service oriented paradigms. An important question in that respect is what EA researchers should do, as opposed to EA practitioners.

2 Contributions

Building on its great success in previous years, the Third Workshop on Trends in Enterprise Architecture Research (TEAR2008) was held in conjunction with the

G. Feuerlicht and W. Lamersdorf (Eds.): ICSOC 2008, LNCS 5472, pp. 313 – 315, 2009.

ICSOC conference in Sydney on December 1st 2008. The TEAR 2008 call for paper attracted 16 submissions. A total of eight papers passed the review process successfully, resulting in a 50% acceptance rate.

The accepted papers reflect the developments in the field of Enterprise Architecture as sketched in the introduction.

Two papers focused on the relationship to Service-Oriented Architectures. The paper "Towards a Sophisticated Understanding of Service Design for Enterprise Architecture" by Stephan Aier and Bettina Gleichauf discusses different service categories, which can be identified in an Enterprise Architecture. Governance aspects with special attention to Service-Oriented Architectures are highlighted by Jan Bernhardt and Detlef Seese in their contribution "A Conceptual Framework for the Governance of Service-Oriented Architectures".

Two submissions reflected usage potentials and implications of Enterprise Architecture in general. Oliver Holschke, Per Närman, Waldo Rocha Flores, Evelina Eriksson, and Marten Schönherr are "Using Enterprise Architecture Models and Bayesian Belief Networks for Failure Impact Analysis". Another usage is shown by Jakob Raderius, Per Närman, and Mathias Ekstedt in their paper "Assessing System Availability Using an Enterprise Architecture Analysis Approach".

Extensions to the understanding of Enterprise Architecture are given in three other contributions. Sabine Buckl, Alexander Ernst, Florian Matthes, and Christian M. Schweda extend the current understanding of Enterprise Architecture with aspects of time in their paper "An Information Model for Landscape Management Discussing Temporality Aspects". The model perspective of managing an Enterprise Architecture is highlighted by Henk Koning, Rik Bos, and Sjaak Brinkkemper in their paper "A Lightweight Method for the Modelling of Enterprise Architectures Introduction and Usage Feedback". The authors pay special attention to keep Enterprise Architectures models as simple a possible in order to increase their net benefit. Christian Riege and Stephan Aier extended the understanding of Enterprise Architectures with situational aspects in their paper "A Contingency Approach to Enterprise Architecture Method Engineering". Finally Marten Schönherr sums up latest developments between individual research groups and practitioners with his paper "Towards a Common Terminology in the Discipline of Enterprise Architecture".

3 Results of the Workshop

The discussion at the workshop resulted in the following questions for further research:

- To what extent is there consensus among enterprise architecture researchers? What are the paradigmatic assumptions on which we all agree?
- Trend 1 in EA research: Decision making and prediction. There seems to be a general consensus that the increased use of EA models for prediction and decision making is a trend for the future.
- Trend 2 in EA research: Complexity of EA models and methods and effort of modelling. It is believed that the problems of keeping the EA program on an acceptable effort and complexity level are becoming increasingly noticed. Research on means for reducing this complexity is therefore considered as a future trend.

There was also a discussion on the future of TEAR. It was agreed that a fourth TEAR workshop should be held. Stephan Aier accepted on behalf of St. Gallen to remain as co-organizer for 2009. A second organizing institution solicited. Marten Schönherr at TU Berlin/T-Labs indicated preliminary interest.

4 Programme Committee

The Programme Committee members and reviewers each deserve credit for forming the excellent final programme that resulted from the diligent review of the submissions. The organizing members of the programme committee would like to thank all of them and especially all authors submitting contributions to the TEAR workshop series.

Members of the Programme Committee

- Stephan Aier, University of St. Gallen, Switzerland (workshop organizer)
- Guiseppe Berio, University of Turin, Italy
- Scott Bernard, Carnegie Mellon University, Syracuse University, USA
- Udo Bub, Deutsche Telekom Laboratories, Germany
- Luis Camarinha-Matos, UNINOVA, Portugal
- Haluk Demirkan, Arizona State University, USA
- Andreas Dietzsch, Postfinance, Bern, Switzerland
- Peter Emmel, SAP, Germany
- Mathias Ekstedt, KTH, Sweden
- Ulrich Frank, University of Duisburg-Essen, Germany
- Matthias Goeken, Frankfurt School of Finance and Management, Germany
- Jan Goossenaerts, Technische Universiteit Eindhoven, The Netherlands
- Michael Goul, Arizona State University, USA
- Pontus Johnson, KTH Stockholm, Sweden (workshop organizer)
- Dimitris Karagiannis, University of Vienna, Austria
- Marc Lankhorst, Telematica Instituut, Enschede, The Netherlands
- Tim O'Neill, University of Technology, Sydney
- Erik Proper, Radboud University Nijmegen and Capgemini, The Netherlands
- Gerold Riempp, European Business School (EBS), Germany
- Michael Rosemann, QUT, Australia
- Joachim Schelp, University of St.Gallen, Switzerland (workshop organizer)
- Marten Schönherr, TU Berlin, Germany
- Gerhard Schwabe, University of Zurich, Switzerland
- Markus Strohmaier, University of Toronto, Canada
- José Tribolet, University of Lisbon, Portugal
- Hongbing Wang, Nanjing University, China
- Alain Wegmann, EPFL, Switzerland
- Martin Zelm, CIMOSA, Germany
- Michael zur Mühlen, Stevens Institute of Technology, USA

Stephan Aier, Pontus Johnson, Joachim Schelp

Towards a Sophisticated Understanding of Service Design for Enterprise Architecture

Stephan Aier and Bettina Gleichauf

Institute of Information Management, University of St. Gallen,
Müller-Friedberg-Strasse 8
9000 St. Gallen, Switzerland
{stephan.aier,bettina.gleichauf}@unisg.ch

Abstract. The service orientation approach emerged form the software engineering community and has now become a widely discussed design paradigm for almost every aspect of an enterprise architecture (EA). However, experience from cases studies has shown that it is necessary to explicitly differentiate service categories in EA, its goals and its resulting design guidelines. This paper derives a sophisticated understanding of different services categories, their respective goals and design guidelines based on empirical research.

1 Introduction

Enterprise architecture (EA) describes the fundamental structure of an organization [22, 25, 29, 33] and supports an organization's transformation by offering a holistic perspective on their elements and the relations between these elements [18]. EA is widely accepted as an approach to manage transformations by (a) propagating strategy and organizational changes consistently to software and infrastructure descriptions, by (b) supporting consistent business transformation enabled by technology innovations, and by (c) decoupling business-oriented and technology-oriented partial architectures [1, 9, 13, 23, 30, 32]. The main challenges of this transformation are the complexity of existing structures, the interdependencies of the architectural elements as well as the heterogeneity of ownership of architectural elements within an organization. The goals that are addressed by EA programs aiming at these challenges are transparency, consistency, IT/Business alignment, stakeholder orientation, standardization, reuse and eventually flexibility and sustainability.

Service orientation as a paradigm not only for software engineering but for enterprise architecture engineering addresses these goals and challenges [26, 27]. However, service orientation precisely discussed in software engineering, e.g. [6, 10, 34], has developed to a broad discussion concerning almost all architectural layers from strategy and business architecture to software, data and infrastructure architecture [33]. Unfortunately this discussion often lacks the necessary precision e.g. when discussing non-technical aspects of service orientation.

When service orientation is discussed as a design paradigm for EA it is indispensable to explicitly differentiate the design goals and their resulting design principles for different design objects in an EA since experience from case studies shows that there is hardly any "one-size-fits-all" design methodology that is equally effective for

G. Feuerlicht and W. Lamersdorf (Eds.): ICSOC 2008, LNCS 5472, pp. 316–326, 2009.

e.g. the design of technology related services as well as business architecture related services at the same time.

Therefore the research questions of this paper are: (a) Which kinds of services should be differentiated from an EA perspective? (b) Which goals are related to these service categories and what are the consequences for service design? To answer these questions we outline three industry case studies to find similarities in their understanding and definitions of services. Based on these cases studies we derive our understanding of services categories in an EA as well as the related goals and design principles for each category.

2 Related Work

Within the last years services were mainly discussed with regard to the technical design of (software) services. There is a multitude of literature concerning this topic and dealing with the actual realization of service orientation, e.g. in the form of web services (e.g. [6, 10, 34]). Besides the realization through software services several authors also claim a comprehensive understanding of service orientation on business layers to the point of a service oriented enterprise [7, 8, 27]. The term "enterprise services" is often used in this context. However, until today there is no consensus either on a clear definition and conceptual delimitation of such services on business layers opposed to software services or among one another or on possible subtypes of these services.

Literature sources say little on the differentiation between the goals that are pursued by the different types of services. Among the general goals mentioned in connection with software services are improvement of process support and increased process flexibility [6], agility of the information systems [11], cost savings due to low maintenance IT systems as well as higher quality due to the possible reuse of software components [7, 16]. In terms of the design of services reuse is acknowledged as the main purpose [14]. LEGNER and HEUTSCHI further identify the goals standardized integration infrastructure and decoupling of application domains [19]. This shows the tight linkage of service orientation and EA. Service orientation can serve as a common "language" between IT and business as it is a fundamental paradigm [18]. The main impact of SOA on EA is the enhanced flexibility in terms of new business partners, new processes, new assets, all while not affecting business operations or enforcing the implementation of a new EA model. Despite these obvious benefits of software service design, there is still no evidence on how these goals can be applied to develop guidelines for the design of services on the business layers of an EA.

With regard to service design most authors refer to the significance of interface orientation and, in conjunction, of service interoperability [11]. Additionally, LEGNER and HEUTSCHI identify the design principles of autonomy and modularity as well as business suitability from existing SOA research on the design of software services [19]. Several authors have presented ideas how enterprise services in particular can be identified and designed [15, 24, 27] but they still remain very close to the concept of software services that are used in a business context. Therefore they lack a business perspective on service design for a comprehensive EA. SCHELP and WINTER show that the conceptual design of enterprise services does not differ considerably from

traditional application design. However they point out that a well-founded and differentiated design method for enterprise services is still missing [28].

Considering the diverse focus of the different service types within a comprehensive EA, the need for differentiated guidelines for services construction becomes evident: In order to adequately apply the required business perspective to business architecture related services and effectively pursue the resulting goals, the construction of such services will differ substantially from software service design. Corresponding design guidelines should be derived from the goals pursued, which have to be identified and structured first. Therefore the paper at hand examines the diversity of service types from an EA perspective and shows how they can be designed in a differentiated way.

3 Case Studies

For deriving a set of differentiated types of services, to describe their design goals as well as their design rules this section presents three case studies. The case studies are used to consolidate essential characteristics of the differentiated service definitions observed. Data for the case studies have been collected with two of these companies since 2002/2003 and with the remaining one since 2006. Key stakeholders in IT management, architecture management (i.e. IS and business architects), and Business/IT relationship management have been interviewed. In addition to the interviews regular review meetings have been set up to observe state, development, and architectural issues in the companies involved. The companies participated in long term collaborative research projects in IS integration and enterprise architecture involving ten companies in the period of 2002–2008. The companies chosen for this study have a long term experience with service oriented architectures and have mature architecture management structures in place. Data presented in the case studies below aggregate the research results gained with these companies until summer 2008. Due to company request the case studies have been made anonymous.

3.1 Company A

Company A is an IT service provider for a large banking network in Germany. In its current form the network is the result of several mergers of formerly independent and regionally organized IT service providers. Every formerly independent company had its own evolutionary grown banking solution. However, none of these solutions had a predominant position within the network. Therefore the network decided to implement a new and common system as their core banking solution. The development started in 2002 and was finished in 2005 for the time being. The new system is designed following a service oriented paradigm in order to adapt and to consistently provide the implemented functionality to every partner.

On a business level the enterprise architecture of company A is designed following the network's *process reference model*. The process reference model serves as a structural blueprint for the design of the actual *business processes* consisting of several steps, e.g. choosing a certain product for a credit application. Single steps of a process are designed as *enterprise services*. These enterprise services are eventually implemented as *software services* on a system level in the enterprise architecture.

Each enterprise service, e.g. management of a credit application, may be used in the entire network for a broad range of products. Throughout the network, reuse of enterprise services is explicitly intended. An enterprise service is comprised of a self-contained set of business functionalities and belongs to a specific domain but may also be reused in other domains. Each enterprise service is linked with exactly one software service as a technical implementation. However, software services may be called in different contexts which may result in a different behaviour.

3.2 Company B

Company B is one of the largest insurance companies in Switzerland. They have started their first projects utilizing a service oriented software design at the end of the 1990ies with the introduction of web applications which integrated basic functionalities of the host systems. These early projects aimed at providing functionality over internet technology. Effects like reuse occurred rather accidentally. However, the potential of service oriented design has soon been recognized and resulted in standardization initiatives as well as an embedment in the company's enterprise architecture framework in order to systematically foster reuse and maintainability of services.

Company B differentiates three layers of service orientation in their architecture: A user-access layer, a process layer and a service layer. The service layer contains business activity services which call business object services. Business object services directly access software systems and may run an update on a database record for example. The process layer does not explicitly define process services but it contains business processes, sub-processes as well as detailed workflow definitions. Workflows employ the functional specifications of the business activity services but may also access business object services directly. On the top level access to application's graphical user interfaces is designed by employing access services implemented in e.g. portals.

The variety of possibilities to use the service framework of company B provides a high flexibility and enables the adaptation to a variety of situations. However, it also demands for strong governance in order to preserve the maintainability of such a framework.

3.3 Company C

Company C is a globally operating telecommunications service provider with a large, complex and heterogeneous application landscape. At the end of the last century, corporate management decided to structure the corporate group into four rather independent divisions representing the four strategic business areas. The new structure reduced the overall complexity by reducing the interdependencies between the four divisions on a business layer as well as on a technology layer by definition. At the same time, however, the heterogeneity as well a redundancy between the divisions grew as a result of their new independence. This independence resulted in e.g. inconsistent information about customers where different divisions served the same customer segments with different, division-specific products. As a consequence, divisions have been continually integrated again in order to leverage synergies among them.

Company C primarily focuses the definition of enterprise services as a solid and standardized definition of business functionalities. The major goal of this standardization is to

provide a reusable repository of enterprise services in order to enable the flexible and fast definition of new or changed products. Consequently the initial identification of enterprise services will be derived from product definitions, e.g. an availability check for an internet connection. For the actual execution the enterprise services employ software services which implement the necessary functionality, e.g. the measurement of signal quality on the physical wire. The encapsulation of required functionality in enterprise services provides the necessary decoupling of the product layer and the technical software layer.

4 Differentiating Services and Guidelines for Service Construction

Contemplating these case studies two categories of services can clearly be identified: On a system layer *software services* carry out the technical realization of functional tasks while *enterprise services* (or business activity services in company B) encapsulate functionalities. At this point it is crucial to determine that the term enterprise services does not stand for pieces of software—as opposed to the wording e.g. in [24]—but it denotes model based abstractions. Furthermore the arrangement of services in the case studies leads to the assumption that there is another relevant layer above the enterprise services. Processes and products are located on this layer and are being served by enterprise services (companies A and C).

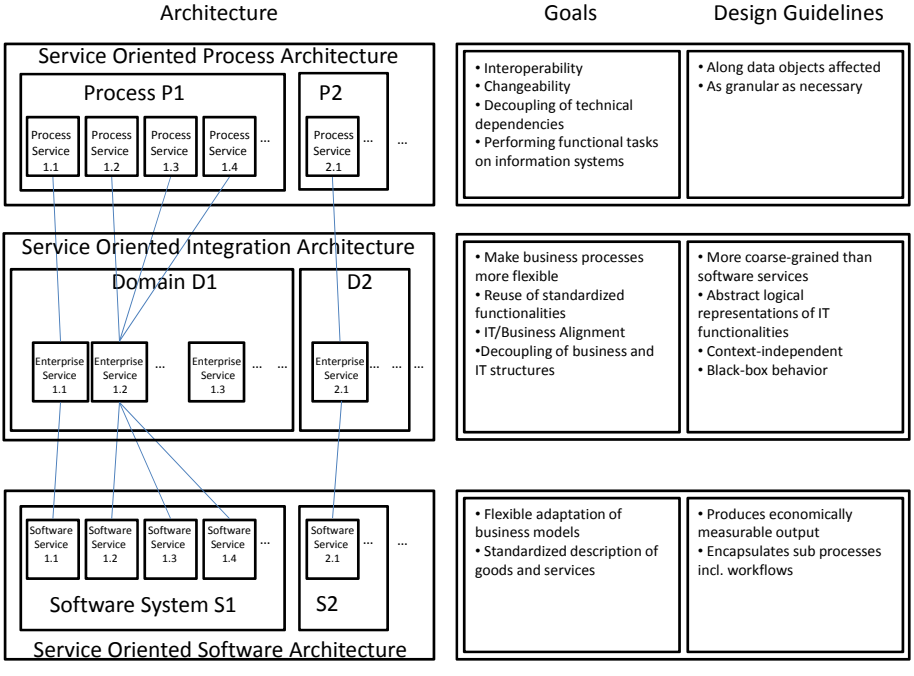


Fig. 1. System of differentiated services following [2]

Taking into account these three layers described by the case studies and an enterprise architecture described by [33], three sub architectures can be distinguished: a *Service Oriented Software Architecture (SOSA)* on the basis of *software services*, a *Service Oriented Integration Architecture (SOIA)* on the basis of *enterprise services* and a *Service Oriented Process Architecture (SOPA)* on the basis of *process services* (Figure 2).

4.1 Service Oriented Software Architecture

The goals of a SOSA are interoperability and changeability on the software layer by the means of software services. Changeability and flexibility should be achieved by decoupling, i.e. by the reduction of technical dependencies. Moreover, software services in a SOSA are used to perform functional tasks on information systems [3].

In order to realize decoupling, the design of software services should be orientated to the data object produced, changed or consumed by them. This is also done in practice like all of the three case studies show: Software services are designed in order to conduct functional tasks on the supporting information systems. There already exists a lot of literature on how to design such software services (cf. chapter 2). According to the goals of a SOSA loose coupling and interoperability are the main guidelines for software services design. Concerning the granularity of software services, our research findings show that services should be as granular as necessary in order to support the functional tasks.

Depending on the task to be performed component services (context oriented services), process services, entity services (data oriented services) and utility services can be identified [24]. For their SOA blueprints OASIS distinguishes between atomic, independent services and composite services. Regarding their statefulness and specific task conversational services, data services, publish-subscribe services and service brokers can be named [21].

The adoption of software services is reasonable in highly volatile environments because interoperability and loose coupling provide advantages in implementation time [16]. On the other side, one finding from our research was that the effectiveness of service orientation in areas that claim extremely high standards concerning performance, transactionality or data security have to be analyzed in detail first (cf. also security and transaction issues discussed in [20]).

4.2 Service Oriented Integration Architecture

A SOIA aims at the creation of enterprise services in order to orchestrate business processes more flexibly. Besides flexibility reuse of standardized functionalities is named as a key objective in the case studies presented above. Company C indirectly aims at process flexibility by concentrating on products as a result of business processes. SCHELP and WINTER [28] mention the connection of business-oriented artifacts and IT artifacts as the main objective of service orientation on the integration layer. Thus, enterprise services aim at supporting IT/Business Alignment. With regard to the alignment objective of enterprise services, case study C seems especially interesting: Enterprise services are applied in order to establish a stable “middle layer” that is able

to realize the relations between the fast changing products on one side and the long-term IT systems on the other side. Besides flexibilisation, reuse and IT/Business Alignment SOIA focuses on the decoupling of different change frequencies of business and IT structures.

In order to operationalize these objectives, enterprise services should provide a discontinuation of interdependencies between functional and technical structures. Therefore, enterprise services should be more coarse-grained compared to the software services underneath and the process services above [2]. Company C is realizing this principle. In company A each enterprise service is assigned to a software service, though the access is possible through different call options. Thus the coarsening between software services and enterprise services is implemented indirectly.

Enterprise services can be defined as abstract logical representations of IT functionalities on the integration layer [28]. To enable flexibilisation of functionalities within business processes, enterprise services should encapsulate coarse-grained functionalities [2]. In contrast to activities enterprise services should be able to exist without contextual information on business processes [31], thus they should not contain information on workflows or the like. Another crucial service design principle in view of the IT/Business Alignment and the flexibilisation of processes is a black box behavior of the services.

KOHLMANN and ALT [15] as well as SCHELP and WINTER [27] present approaches for the design of enterprise services: KOHLMANN and ALT [15] propose a business-driven identification of enterprise services by deducing them from reference processes while considering role models. The orientation towards processes, functions and data objects as well as the aggregation of self-contained activity sets is also named by SCHELP and WINTER [27] among the main design guidelines for enterprise services. This approach seems to be applicable in practice, too: Company A identifies enterprise services by the means of given reference processes while company B is deducing the specification for enterprise services directly from their business activities and business objects that have to be realized.

Enterprise services can be differentiated in loose coupled, i.e. orchestrated, or tightly coupled, i.e. composed services [27]. KOHLMANN and ALT [15] distinguish between enterprise service types for process activities, business rules and business objects. Company B also models activity oriented and business object oriented services. According to the traditional application design SCHELP and WINTER [28] distinguish product centered, information centered and function centered enterprise services. In practice the differentiation along contextual factors seems sensible: In company A enterprise services are assigned to application domains so the services are grouped by explicit business contexts. This may lead to the differentiation in internal or external services.

In order to successfully apply service orientation to functionalities the decomposition of functionalities should be logically possible. The presented companies A, B and C only partition and encapsulate functionalities if they affect multiple business objects, like customers or business processes, so that reuse is possible. If the decomposition in services results in high complexity and reuse potentials are no longer given, service orientation should not be applied in this business area. For example Company A does not design singular functionalities by the service orientation paradigm.

4.3 Service Oriented Process Architecture

On the organization or process layer, service orientation can be used to flexibly adapt business models by the means of process services. An example for the need of such an adaptation is the outsourcing of individual business functions or processes (business process outsourcing). The essential explication of service interfaces makes the shift of goods and services into another environment possible. By defining process services a structured description of goods and services is achieved at the same time.

Process service should encapsulate sub processes respectively a self-contained activity set that has an economically measurable output. Opposed to enterprise services process services also incorporate sub processes' workflows that are deduced from existing process models that exist in the company. In order to anticipate the possible changes in the process design, like outsourcing or inter-corporate collaboration, the design of process services should consider the usage of non-exclusive, common standards [4].

The delimitation of process services as opposed to enterprise services on the integration layer seems to be tough in practice, as our case studies show: In company A individual process steps are identified but not explicitly implemented as a service. Company B differentiates between business activities and business objects but as regards content they are rather instances of enterprise services than process services.

In view of the flexibility for new business models the implementation of process services seems reasonable if the sub processes can be standardized concerning their quality and execution.

5 Discussion

This contribution illustrates the necessity to precisely and explicitly differentiate service categories from an EA perspective. The main reasons to differentiate services are the different goals and resulting different design guidelines applying to these service categories.

Based on three cases studies we differentiate software services, enterprise services and process services. While the cases studies and also literature provide a profound understanding of software services and enterprise services, the design of process services is dominated by a classical top-down process design. The fact that classical business process management and process modelling predominantly use a top-down approach while service orientation traditionally aims at clustering existing assets applying a bottom-up methodology may lead to new challenges but also to new synergies. In process modelling research, reuse of process building blocks has also been a topic discussed for several years, often in conjunction with (process) reference models (e.g. [5, 17]). It can be supposed that such approaches are relevant and fruitful for process service design. Therefore, future research should elaborate on the appropriateness of "classical" process design in a services oriented EA. In doing so, the applicability and the practical use of SOPA can be further evaluated.

An opportunity for further research can also be found in the field of service design guidelines. Neither literature nor the practitioner's community provide an empirically validated understanding of which design guidelines effectively contribute to their

respective goals. There are only a few publications on service design guidelines—and only as far as IT architecture is concerned. Regarding integration and process architecture, there is hardly any publication on service design. Empirical validation of successful service design guidelines—especially for service oriented integration and process architecture—is non-existent and practitioners' experiences are limited. However, service design guidelines which have proven to be successful will be beneficial for the practitioners' community and therefore represent a reasonable goal for explorative research.

We strongly encourage research which investigates the influence of contextual factors on the success of services orientation. The contingency theory of leadership [12] argues that there is no "best way" of organizing or leading an organization. Instead, there are various internal and external factors that influence organizational effectiveness. The organizational style must therefore be contingent upon those factors. Translated into the context of success of service orientation, it stands to reason that contingency factors such as industry sector, organizational size or prior experience with service orientation might influence SOA success.

Finally, for a validation of the concept of different service sub-architectures, the interplay of SOPA, SOIA and SOSA should be analyzed in detail. It can be assumed that the relationships between the different services on different layers are quite complex—both, as EA models and as their observable artifacts. In order to describe and to establish a comprehensive and flexible EA based on services, the links between the different layers need to be understood thoroughly.

References

1. Aier, S., Riege, C., Winter, R.: Classification of Enterprise Architecture Scenarios – An Exploratory Analysis. Enterprise Modelling and Information Systems Architectures 3(1), 14–23 (2008)
2. Aier, S., Winter, R.: Virtual Decoupling for IT/Business Alignment – Conceptual Foundations, Architecture Design and Implementation Example. Business & Information Systems Engineering 51(2) (forthcoming, 2009)
3. Alonso, G., Casati, F., Kuno, H., Machiraju, V.: Web Services – Concepts, Architctures and Applications. Springer, Heidelberg (2004)
4. Alt, R., Fleisch, E.: Key Success Factors in Designing and Implementing Business Networking Systems. In: Proceedings of Proceedings of 12th Electronic Commerce Conference: Global Networked Organizations, Bled, Slovenia, Kranj, Slovenia, Moderna organizacija, June 7–9, 1999, pp. 219–235 (1999)
5. Baacke, L., Rohner, P., Winter, R.: Aggregation of Reference Process Building Blocks to Improve Modeling in Public Administrations. In: Proceedings of Electronic Government, 6th International EGOV Conference, Proceedings of ongoing research, project contributions and workshops, Trauner Druck, pp. 149–156 (2007)
6. Berbner, R., Grollius, T., Repp, N., Eckert, J., Heckmann, O., Ortner, E., Steinmetz, R.: Management of Service-oriented Architecture (SOA)-based Application Systems. Enterprise Modelling And Information System Architectures 2(1), 14–25 (2007)
7. Bieberstein, N., Bose, S., Walker, L., Lynch, A.: Impact of Service-oriented Architecture on Enterprise Systems, Organizational Structures, and Individuals. IBM Systems Journal 44(4), 691–708 (2005)

8. Brown, G., Carpenter, R.: Successful Application of Service-Oriented Architecture Across the Enterprise and Beyond. Intel Technology Journal 8(4), 345–359 (2004)
9. Buchanan, R.D., Soley, R.M.: Aligning Enterprise Architecture and IT Investments with Corporate Goals, Object Management Group (2002)
10. De Backer, M., Dedene, G., Vandelbulcke, J.: Web Services Composition, Execution and Visualization. In: Proceedings of Proceedings of the 12th IEEE International Workshop on Program Comprehension (IWPC 2004), pp. 264–265. IEEE Computer Society Press, Los Alamitos (2004)
11. Erl, T.: SOA: Principles of Service Design. Prentice Hall, Upper Saddle River (2007)
12. Fiedler, F.E.: A Contingency Model of Leadership Effectiveness. Advances in Experimental Social Psychology 1, 149–190 (1964)
13. Fischer, R., Aier, S., Winter, R.: A Federated Approach to Enterprise Architecture Model Maintenance. Enterprise Modelling and Information Systems Architectures 2(2), 14–22 (2007)
14. Herr, M., Bath, U., Koschel, A.: Implementation of a Service Oriented Architecture at Deutsche Post MAIL. In: Zhang, L.-J., Jeckle, M. (eds.) ECOWS 2004. LNCS, vol. 3250, pp. 227–238. Springer, Heidelberg (2004)
15. Kohlmann, F., Alt, R.: Business-Driven Service Modeling – A Methodology Approach from the Finance Industry. In: Proceedings of BPSC 2007: Proceedings of the 1st International Working Conference on Business Process and Services Computing, Leipzip, pp. 180–193 (2007)
16. Krafzig, D., Banke, K., Slama, D.: Enterprise SOA – Service-Oriented Architecture Best Practices, The Coad Series. Pearson, Upper Saddle River (2005)
17. Lang, K., Bodendorf, F., Glunde, J.: Framework for Reusable Reference Process Building Blocks. SIGGROUP Bulletin 18(1), 68–70 (1997)
18. Lankhorst, M.: Enterprise Architecture at Work: Modelling, Communication and Analysis. Springer, Berlin (2005)
19. Legner, C., Heutschi, R.: SOA Adoption in Practice – Findings from Early SOA Implementations. In: Proceedings of Proceedings of the 15th European Conference on Information Systems, Relevant rigour – Rigorous relevance (2007)
20. McGovern, J., Sims, O., Jain, A., Little, M.: Enterprise Service Oriented Architectures: Concepts, Challenges, Recommendations. Springer, Dordrecht (2006)
21. OASIS, SOA Blueprints, OASIS (2005) (last access: 08.08.2008), http://www.oasis-open.org/committees/download.php/15965/05-12-00000.001.doc
22. Rood, M.A.: Enterprise Architecture: Definition, Content, and Utility. In: Proceedings of Proceedings of the Third Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, pp. 106–111. IEEE Computer Society Press, Los Alamitos (1994)
23. Ross, J.W., Weill, P., Robertson, D.C.: Enterprise Architecture as Strategy. Creating a Foundation for Business Execution. Harvard Business School Press, Boston (2006)
24. SAP, Enterprise Services Design Guide, SAP AG (2006)
25. Schekkerman, J.: How to Survive in the Jungle of Enterprise Architecture Frameworks: Creating or Choosing an Enterprise Architecture Framework, 2nd edn. Trafford Publishing, Victoria (2004)
26. Schelp, J., Aier, S.: SOA and EA – Sustainable Contributions for Increasing Corporate Agility. In: Proceedings of 42th Hawaii International Conference on Systems Science (HICCS-40). IEEE Computer Society Press, Los Alamitos (2009)

27. Schelp, J., Winter, R.: Towards a Methodology for Service Construction. In: Proceedings of 40th Hawaii International Conference on Systems Science (HICCS-40). IEEE Computer Society, Los Alamitos (2007)
28. Schelp, J., Winter, R.: Business Application Design and Enterprise Service Design: A Comparison. Int. J. Service Sciences (2008)
29. The Open Group, The Open Group Architecture Framework TOGAF – 2007 Edition (Incorporating 8.1.1), Van Haren, Zaltbommel (2007)
30. Veasey, P.W.: Use of Enterprise Architectures in Managing Strategic Change. Business Process Management Journal 7(5), 420–436 (2001)
31. Vernadat, F.B.: Reengineering the Organization with a Service Orientation. In: Hsu, C. (ed.) Service Enterprise Integration. Integrated Series In Information Systems, vol. 16, pp. 77–101. Springer, Heidelberg (2007)
32. Wagter, R., van den Berg, M., Luijpers, J., van Steenbergen, M.: Dynamic Enterprise Architecture: How to Make It Work. John Wiley & Sons, Hoboken (2005)
33. Winter, R., Fischer, R.: Essential Layers, Artifacts, and Dependencies of Enterprise Architecture. Journal of Enterprise Architecture 3(2), 7–18 (2007)
34. Yuan, R., Zunchao, L., Boqin, F., Jincang, H.: Architecture-based Web service composition framework and strategy. In: Proceedings of 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS 2005), pp. 129–134. IEEE Computer Society, Los Alamitos (2005)

A Conceptual Framework for the Governance of Service-Oriented Architectures

Jan Bernhardt[1] and Detlef Seese[2]

[1] SAP AG, Dietmar-Hopp-Allee 16, 69190 Walldorf, Germany
[2] Institute AIFB, University of Karlsruhe, 76128 Karlsruhe, Germany
jan.bernhardt@sap.com, seese@aifb.uni-karlsruhe.de

Abstract. With the widespread adoption of service-oriented architecture (SOA) as the leading paradigm for the development of business applications, the need for adequate governance mechanisms to keep control of the increasing complexity inherently introduced by SOA arises. However, current approaches to SOA governance are often limited to either design time or runtime aspects of SOA governance, despite the need for adequate governance mechanisms covering the complete SOA lifecycle. Furthermore, no common understanding of SOA governance exists today, as many solution vendors misleadingly label SOA management products as solutions for SOA governance.

This work presents a reference model for SOA governance that is based on the OASIS Reference Model for Service Oriented Architecture. It aims at establishing a common understanding of SOA governance and is intended to serve as a basis for comparing and developing concrete SOA governance solutions.

1 Introduction

To date neither a common understanding of, nor a common model for SOA governance exists. Moreover, current approaches proposed by SOA governance solution vendors are incomplete in that they only address some aspects of SOA governance, mainly technical governance and management aspects. Our reference model for SOA governance strives at closing these gaps. It is based on a reference model for SOA methodologies (see [1]), which in turn is based on the OASIS Reference Model for Service Oriented Architecture (SOA-RM) [2].

In order to account for the special nature of SOA – increased complexity (e.g. [3]), augmented security requirements (cf. [1]), and demand for better business and IT alignment (e.g. [4]) – the discipline of SOA governance has evolved. It is an extension or evolution of IT governance defining the organizational structures and processes that are required to successfully adopt SOA (e.g. [4], [5]).

Drawing on [6] and on definitions of corporate [7] and IT governance [8] we define SOA governance as follows:

Definition 1 (SOA governance). *SOA governance consists of the organizational structures, processes, and policies an organization puts in place to ensure*

G. Feuerlicht and W. Lamersdorf (Eds.): ICSOC 2008, LNCS 5472, pp. 327–338, 2009.

that the adoption, implementation, and operation of SOA in an organization is done in accordance with best practices, architectural principles, government regulations, and laws, that is, in a manner that sustains and extends the organization's strategies and objectives.

The SOA-RM defines SOA as "a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains" providing "a uniform means to offer, discover, interact with and use capabilities to produce desired effects" [2]. Based on this understanding of SOA and taking into account its special nature we have developed a reference model for SOA governance consisting of a comprehensive set of governance policies, processes, and corresponding organizational constructs.

The remainder of this work is organized as follows. Sect. 2 summarizes related work in the area and identifies gaps in current approaches, motivating the need for a reference model for SOA governance. The reference model is presented in Sect. 3. We conclude our work in Sect. 4 by giving an outlook on areas for future research.

2 Related Work

Over the last few years SOA governance has been a topic primarily driven by analysts (e.g. [6], [9], [10]) and technical governance and management solution vendors (e.g. [11]), and only recently scientific literature has been published in the area (e.g. [5]). In addition, some of the recent books on SOA also contain chapters about SOA governance (e.g. [4], [12]). Finally, numerous online articles are available covering the topic in more or less detail (e.g. [13], [14], [15]).

Some main characteristics of SOA governance recur throughout the literature. First if all, the need for SOA governance is motivated from the increased complexity of SOA originating from the number of services and their interdependencies that exist in a SOA landscape (e.g. [4], [5], [12]). Secondly, consensus seems to exist about the fact that SOA governance can be seen as a specialization (or extension) of IT governance accounting for the special nature of SOA (e.g. [4], [12], [14], [15]). Thirdly, [4], [5], and [6] name policies, processes, organizational structures, and metrics as core constituents of a SOA governance program. In addition it is stated that in order to successfully fulfill its mandate SOA governance needs control over project approval and funding as well as portfolio management decisions (e.g. [4], [13]). Finally, reuse of services is commonly named as a main benefit of SOA (e.g. [9], [13]). However, [5] legitimately argues that reuse contradicts agility.

Based on the SOA-RM [2] OASIS has published a Reference Architecture for Service Oriented Architecture [16], which contains a generic model for governance augmented with some specifics for SOA governance. Furthermore, The Open Group has set up a project on SOA governance [17] as part of its SOA working group, whose aims include the definition of a reference model for SOA governance. However, no results have been published to date.

What distinguishes the reference model presented in this work from available material in the domain is the rigorously holistic approach assumed, covering the complete SOA lifecycle from service proposition to retirement. Moreover, all aspects that should be considered by SOA governance have been identified from a reference model for SOA methodologies (see [1]) and are presented in a comprehensive, uniform set of governance policies, processes, and organizational constructs, which to the best of our knowledge is unprecedented in the literature.

3 A Reference Model for SOA Governance

In order to successfully adopt SOA in an organization, certain organizational structures, consisting of a number of organizational units, have to be established to govern the introduction and operation of SOA. The organizational units define governance policies and processes, execute governance processes, and employ metrics to measure adherence to them (Fig. 1).

Governance processes enable governance policies and enforce compliance. Metrics provide visibility into the processes allowing for supervision, control, and identification of areas for improvement. Furthermore, a SOA governance infrastructure helps to manage governance information, enforce governance policies, and collect corresponding metrics (cf. [6]). The following subsections describe the proposed constituents of SOA governance in detail.

3.1 Organizational Structure

The organizational structure for SOA governance comprises the SOA governance board, the SOA program office, and a number of delegate councils.

SOA Governance Board. The SOA governance board constitutes the main governance body of a SOA initiative. It should be composed of specialists from both the business and technology domains such as business analysts, enterprise architects, and service architects. In addition, it is important that decision makers such as the line of business managers that are affected by the SOA initiative

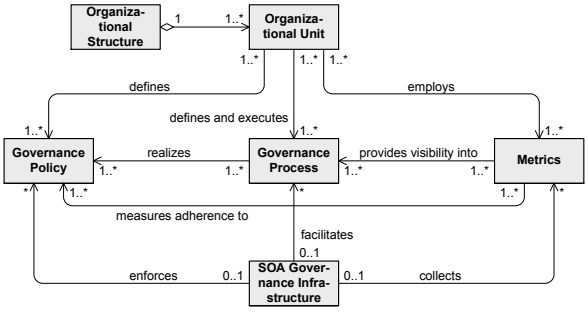


Fig. 1. Elements of SOA governance

are also part of the governance board to establish the required influence and authority.

In particular, the SOA governance board takes into account business objectives, architecture and design standards and best practices, and applicable laws to translate them into corresponding governance policies. It also defines metrics that will be used to measure adherence to and success of the policies. Furthermore, governance processes have to be defined and set up that integrate with the SOA lifecycle to realize the policies.

The governance board is supported by a number of councils to which it may delegate work (cf. [16], see [1] for details). Depending on whether an organization applies a central or distributed SOA governance scheme (cf. [4]), the board either possesses exclusive decision rights regarding policies and processes, or it may empower the delegate councils to make decisions themselves.

As a transition to SOA is a highly complex undertaking with impact on many different parts of an organization, the transition should take place iteratively (cf. [4]). It is the responsibility of the SOA governance board to set up an overall SOA roadmap that accounts for progressive adoption of SOA.

Finally, project prioritization and funding are necessary means to steer the SOA initiative into the desired direction, particularly with respect to business alignment of services. Without practical control of funding projects are more likely to produce project-centric services which might not meet the overall requirements of the organization [13]. Therefore the SOA governance board must at least possess authority to influence decisions on project prioritization and funding.

SOA Program Office. The SOA program office is the main organizational management body in a SOA initiative, which is in charge of the overall SOA program management. Its responsibility is to set up the required organizational structures, first and foremost the delegate councils.

The program office reports to the SOA governance board. It is authorized to decide on project prioritization, selection and funding in accordance with the board, which may take influence in the decision process. The program office also assigns people to the roles defined by the governance board. Finally, it may also influence the SOA roadmap.

3.2 Policies

Policies are the means to define what is *right*. This reference model distinguishes two notions of policies. Governance policies, on the one hand, formulate general rules that address important aspects of SOA such as how to develop, introduce and operate a SOA. Governance policies state on a high level what to do and how to do it. Service policies, on the other hand, are more concrete; they directly affect, and can be attached to one or more services.

Definition 2 (Governance policy). *Governance policies comprise rules that on a high level address all aspects that are important in developing, deploying*

and managing a SOA, that is, the SOA lifecycle. Governance policies define what to do and how to do it.

Definition 3 (Service policy). *Service policies define rules that specifically apply to one or more services. They are always derived from governance policies.*

Service and governance policies are both specializations of policies. A policy consists of one or more policy assertions, the fulfillment of which is measurable [2]. When an individual or an organization adopts one or more policy assertions as their policy, the policy assertions become the policy of that individual or organization. In the scope of this reference model we do not explicitly distinguish between policies and policy assertions, but reluctantly speak of policies instead.

Every policy has a dedicated policy owner. The policy owner is the one who has an interest in the policy. He usually is responsible for the enforcement of the policy [2].

In our work we have identified 41 different governance policies for the complete SOA lifecycle (i.e. service proposition, design, implementation, provisioning, consumption, and management). These policies are motivated from the definition of SOA given in [2]. They ensure that all activities associated with the SOA lifecycle are executed in a manner that sustains the organization's objectives. In the following paragraphs we present the most important of the identified governance policies; for a full description see [1].

Service Proposition. For service proposition the approach that should be taken has to be specified (service proposition policy). Services can either be proposed in a top-down, bottom-up, or in a combined manner (cf. [18], [19]). Top-down service proposition identifies service candidates from some kind of business model, whereas bottom-up service proposition uses existing application system functionality to propose services. It is also possible to apply a combination of the two alternatives.

Furthermore, fundamental choices regarding general service-oriented design principles, which also have an effect on service proposition, have to be made. Two aspects that need to be considered in this regard are service granularity and reusability (e.g. [20]; service granularity and reusability policies). Designing services for reuse is more expensive [5], wherefore the benefit of avoiding duplicate functionality has to be weighed against the additional costs. Moreover, service reuse decreases agility [3], as it increases the number of dependencies that have to be considered in case services need to change. Service granularity and reusability also affect each other, since services with a narrower focus may be more easily reused [20].

Service Design. Regarding service design it has to be determined whether different service types should be distinguished (e.g. infrastructure, data, functional, and process services; cf. [21]). This is stipulated in a corresponding service type policy.

Another fundamental aspect of service design is the use of service design patterns. A number of design patterns for services are emerging (e.g. [22]), and

it has to be determined if any of these patterns should be applied for service design (design pattern policy).

Furthermore, the service error handling scheme has to be determined (error handling policy). Options include using a standard (e.g. SOAP) or a proprietary error handling mechanism, and to handle all types of errors (e.g. application errors, communication errors) in the same way or differently.

Finally, the transaction behavior of services also has to be taken into account (transaction behavior policy). The question to answer here is whether service invocations should adhere to the ACID (atomicity, consistency, isolation, and durability) characteristics common in database design.

For service interface modeling it has to be defined whether service interfaces should be designed according to common message schema standards (e.g. ebXML, RosettaNet). In case many services will be disclosed to external parties, it may be advantageous to comply with common standards. On the other hand, if the number of public services is low, the development of proprietary schemas following the organization's data architecture may be more applicable (message schema policy).

An important aspect of a SOA initiative is the unification of data types across the organization (cf. [23]) to achieve a common understanding of data types, to allow for reuse, to create compatible service interfaces in terms of syntax and semantics, and to keep message transformations during service operations to a minimum (data type unification policy). In addition, data type definition and reuse policies stipulate naming rules for data types, application of common standards for data type definition (e.g. Core Components Technical Specification) to facilitate semantic service interoperability, and rules regulating reuse of data types.

Another fundamental decision refers to transaction patterns (transaction pattern policy). The definition and prescription of transaction patterns such as query/response or request/confirmation for all service interfaces provides a means to unify the semantics of the interfaces, which enhances service interoperability (see e.g. [20]).

Consistent naming of service interfaces, i.e. the definition of naming conventions, is another important aspect (naming policy). In case transaction patterns are employed, it may be beneficial to include the employed pattern into the interface and operation names to clearly indicate their semantics. For example, the interface of a customer management service that adheres to the request/confirmation pattern may have *customerCreateRequest* and *customerCreateResponse* operations.

Finally, it has to be discussed whether reliable communication is required and how it should be achieved (reliable communication policy).

Service Implementation and Quality Assurance. Regarding service implementation governance has to decide whether services should be implemented on the basis of a component technology (e.g. Service Component Architecture (SCA) [24]). Furthermore, the implementation and interface technologies together with the technical frameworks that should be used have to be determined

(service component policy and implementation technology policy, respectively). As it is one of the aims of SOA to allow services built using a variety of platforms and programming languages to interact, it is possible that multiple languages are chosen for implementation depending on the specific requirements.

Governance concerning service quality assurance is a crucial aspect for the success of SOA and ranges from determining tests that have to be conducted (testing policy) to reviews that have to be passed (review policy). Common tests for services include dependency, interoperability, and regression tests for functional service testing, and performance, stress, and security tests for non-functional service testing. In addition, scenario tests to test a number of dependent services in combination are particularly important (cf. [6]).

Service Documentation. The service description is a fundamental part of SOA as it provides information necessary to interact with a service to potential service consumers. A service documentation policy therefore has to ensure that all relevant information is actually included in the service description. It is probably useful to define a service description template in this respect which has to be adhered to. Moreover, with regard to service discovery, it is advisable to define a service taxonomy and to classify all services according to this taxonomy to alleviate discovery of services that can satisfy a consumer's needs.

According to [2] the service description should at least contain information about the functionality of the service, access to the service (i.e. the service address), and policies the service operates under. Concerning the service interface the required message syntax, the underlying semantics, and allowed message sequences, including applicable transaction patterns, have to be specified. In addition, main characteristics of the service such as statelessness, idempotency, and warranties regarding reliable communication should be part of the service description as well.

Finally, service policies such as guaranteed service levels and according remuneration schemes, as well as the service's error handling mechanism should also be described.

Service Publication and Consumption. Regarding service publication it has to be decided where services (i.e. service descriptions) should be published, i.e. if a service registry should be employed, or whether services should simply be published in a file, for example (service publication policy). In case it is decided to use a registry, options include using separate registries for service publication and service governance (e.g. for maintaining administrative information and service lifecycle states), or a single registry for both.

Furthermore, a role concept and according access control mechanisms for the registry have to be determined, including decisions regarding who is allowed to publish and who entitled to release services for production.

Finally, in case a service taxonomy has been defined for services, the attributes assigned to each service should be published in the registry as part of the service description.

The role concept and access control mechanisms for the registry also have to take into account rules for service consumption. The corresponding service consumption policy has to specify who is allowed to search for services, access services (e.g. for testing, for production), and to generate service proxies from the service interfaces in order to consume services.

Another important aspect of service consumption is contract negotiation. Therefore the service consumption policy also has to determine who is responsible for and entitled to negotiating contracts with service consumers. This could, for example, be the service owner.

Finally, the contract language is another important aspect to consider. Depending on the subject of the contract it may be specified that contracts have to be expressed in a machine-processable form to permit automated interpretation and enforcement (cf. [2]).

Service Management. Services in production have to be monitored to ensure that non-functional service requirements specified during service design, and service levels negotiated during service consumption are maintained. Service monitoring inherently is a management activity. It is the responsibility of governance, however, to specify what has to be monitored and to define remedial actions in case violations occur (service monitoring policy). Service monitoring likely includes service level, availability, security, and error monitoring.

Besides, distinct policies define how individual management responsibilities should be handled. A service level management policy defines how contracted service levels should be maintained, remedial actions in case service levels are violated, and different service levels and according pricing models offered.

A service availability policy specifies how service availability management should be performed in order to guarantee the degree of service availability assured to service consumers. This again includes the definition of remedial actions in case availability is threatened.

A security management policy determines how security management should be performed. This may reach from high level governance rules stating that sensitive services have to be secured in some way, to concrete rules prescribing the actual access control and encryption mechanisms to be applied.

Finally, a dependency management policy specifies how service dependencies should be managed. One of the main challenges in SOA is the increased complexity that has to be handled and which stems from the number of interdependent services existing in a SOA landscape together with their organizational structure and the dynamics of their interaction (cf. [5]). If these dependencies are not structured well and managed without gap, service changes may lead to critical errors.

Service Versioning, Change, Deprecation, and Retirement. Due to changing business and other requirements, services are likely to change during their lifetime. Governance has to decide how changes are handled, and whether incompatible or only compatible changes are allowed (change policy).

Closely related to change is the topic of versioning (versionig policy). For example, it could be specified that compatible changes entail a new service version,

whereas incompatible changes require a new service with a different name and different taxonomy attributes. Another aspect pertains to how many versions of a service are allowed to exist in parallel. Increasing maintenance costs probably necessitate the retirement of older versions at some point in time.

Furthermore, service deprecation and retirement procedures, as well as transition periods have to be appointed (service deprecation and retirement policies). Transition periods may be subject to negotiation and codified in the service contract.

Finally, service providers may want to notify service consumers of events such as service change, deprecation, and retirement (notification policy). The obligation to send notifications may also be included in the service contract.

3.3 Governance Processes

Governance processes are the means to realize governance policies. This includes policy-related processes such as definition and enforcement, as well as project-related processes such as prioritization and funding. Furthermore, review processes as an instrument to verify policy compliance are another important aspect of SOA governance (for details see [1]). To minimize the overhead inherently introduced by any governance mechanism, governance processes should be automated wherever feasible.

Policy-Related Processes. A number of governance processes revolve around the definition, propagation, and enforcement of governance policies. Having established an appropriate organizational structure for SOA governance, the respective organizational bodies (i.e. the SOA governance board, the delegate councils) start defining and documenting governance policies for their specific areas of expertise. The policies may have to pass a resolution process, particularly if policies developed by delegate councils have to be ratified by the governance board. Besides, it is necessary to educate on the policies (policy enablement).

Furthermore, mechanisms enabling policy enforcement have to be established. Enforcement can either be achieved manually (e.g. by a design review), or automatically (e.g. by a SOA management system). Closely related to enforcement is the handling of policy violations, which likely comprises initiating compensatory actions (cf. [2]). In addition, policy waiver procedures should be defined (cf. [6]) to be able to bypass certain policies in case special business requirements require to do so.

Finally, policies may have to be adapted due to changing business requirements, or because they prove inadequate in practice. The concept of versioning is therefore applicable not only to services, but to policies as well.

Approval, Review and Reporting. Review processes are a means to enforce governance policies, particularly regarding design time policies. Passing a review may be a prerequisite for the transition to the next phase of the SOA lifecycle (cf. [13]). For example, before moving on to service implementation, service models may have to pass a service design review.

Closely related to reviews are approval processes, which are another means for policy enforcement. In particular, new services or service changes may have to be approved by the SOA governance board before they may be realized. Besides services, other design decisions such as the introduction of new data types may be subject to approval as well.

Finally, reporting processes provide feedback from the organizational units concerned with realizing the SOA to the SOA governance board, which it requires to maintain overview of the SOA initiative and to intervene if necessary.

3.4 Metrics

Metrics improve transparency and provide visibility of essential states and parameters of the SOA initiative on the whole, and into governance processes in particular to the SOA governance board (compare corresponding developments on lower levels such as software or process metrics, see e.g. [25]). They may be partitioned into metrics related to services, service operations and projects.

Regarding services it is interesting to know how many services exist in total, have been proposed, are being developed, are published, and are actually consumed. Furthermore, numbers on changed, deprecated and retired services also provide useful insights. Finally, information on the number of consumers per service and service version is helpful as well, particularly regarding the impact of changes.

To provide visibility into service operations and to identify areas for improvement a number of operations-related metrics are helpful. These include the number of successful vs. the number of erroneous interactions, including information indicating the reasons for the errors (e.g. SOAP faults, invalid input data, size of input data etc.). Another important aspect is performance KPI's, as these help to identify both bottlenecks in the infrastructure and the need for service design revisions. Furthermore, service level and security violations are also important to report.

Finally, in order to maintain the overview of the SOA initiative on the whole it is important to provide feedback on the number of successful and failed projects and the reasons for both failure and success.

3.5 SOA Governance Infrastructure

SOA governance processes must be as unobtrusive as possible to encourage adoption of SOA. A SOA governance infrastructure (SGI) helps facilitating SOA governance by managing governance information, automating governance processes, enforcing governance policies, and collecting metrics (cf. [6]).

The elements of an SGI are determined by the specific governance requirements. A service registry helps to maintain overview of existing services and to manage administrative information such as service lifecycle states. It may also allow for service consumer notifications, e.g. using some kind of publish/subscribe mechanism for change and deprecation events. For the management of design artifacts a repository can be employed. It is also a natural candidate for a central data type inventory.

Policy and contract management are supported by policy and contract management systems. A policy management system helps to define and document policies, attach policies to artifacts, and to propagate policies to their respective points of application. It may also provide capabilities for analyzing the impact of policy changes.

With respect to the elevated testing requirements in SOA a specialized quality management system may support testing complete service compositions, as well as impact analysis in case of service changes, taking into account dependencies between services.

Finally, a SOA management system should facilitate service administration (e.g. lifecycle transitions) and monitoring. Moreover, automatic policy enforcement mechanisms, as well as reporting capabilities should also be provided.

4 Conclusion

This work has outlined a comprehensive reference model for SOA governance that is based on the standardized SOA-RM [2] and motivated from aspects relevant to methodologies for SOA (see [1]). Unlike previous approaches to SOA governance, our reference model addresses governance aspects for the complete SOA lifecycle, stipulated in a comprehensive set of governance policies, processes, and organizational considerations.

Having established a common understanding of the concepts relevant for SOA governance, in a next step we plan to investigate in detail the relationships between our reference model and common frameworks for IT governance (e.g. COBIT) and Enterprise Architecture (e.g. TOGAF, Business Engineering Model [26]).

References

[1] Bernhardt, J.: A Conceptual Framework for the Governance of Service-Oriented Architectures. Master's thesis, Universität Karlsruhe (TH), Germany (April 2008)
[2] MacKenzie, C.M., Laskey, K., McCabe, F., Brown, P.F., Metz, R.: OASIS Reference Model for Service Oriented Architecture 1.0 (October 2006)
[3] Schelp, J., Winter, R.: Towards a Methodology for Service Construction. In: HICSS 2007: Proceedings of the 40th Annual Hawaii International Conference on System Sciences, Washington, DC, USA, p. 64. IEEE Computer Society, Los Alamitos (2007)
[4] Bieberstein, N., Bose, S., Fiammante, M., Jones, K., Shah, R.: Service-Oriented Architecture Compass: Business Value, Planning, and Enterprise Roadmap, 2nd edn. IBM Press Pearson Education, Upper Saddle River (2006)
[5] Schelp, J., Stutz, M.: Serviceorientierte Architekturen (SOA). In: SOA-Governance, 1st edn. ser. HMD – Praxis der Wirtschaftsinformatik, pp. 66–73. dpunkt-Verlag, Heidelberg (2007)
[6] Manes, A.T.: You Can't Buy Governance, InfoWorld SOA Executive Forum (November 2007), http://www.infoworld.com/
[7] Organisation for Economic Co-Operation and Development, "OECD Principles of Corporate Governance (2004), http://www.oecd.org/

[8] IT Governance Institute (ITGI), Board Briefing on IT Governance, 2nd edn. (2003), http://www.itgi.org/
[9] Malinverno, P.: Service-Oriented Architecture Craves Governance (January 2006), http://www.gartner.com/
[10] Malinverno, P.: Sample Governance Mechanisms for a Service-Oriented Architecture (April 2006), http://www.gartner.com/
[11] Systinet, SOA Governance: Balancing Flexibility and Control Within an SOA (September 2006), http://www.hp.com/
[12] Starke, G., Tilkov, S. (eds.): SOA-Expertenwissen: Methoden, Konzepte und Praxis serviceorientierter Architekturen, 1st edn. dpunkt-Verlag, Heidelberg (2007)
[13] Carlson, B., Marks, E.: SOA Governance Best Practices – Architectural, Organizational, and SDLC Implications (January 2006), http://webservices.sys-con.com/
[14] Brown, W.A., Cantor, M.: Soa governance: how to oversee successful implementation through proven best practices and methods (August 2006), http://www.ibm.com/
[15] Starke, G.: SOA Governance: Crucial Necessity or Waste of Time?(November 2007), http://www.infoq.com/
[16] McCabe, F.G., Estefan, J.A., Laskey, K., Thornton, D.: OASIS Service Oriented Architecture Reference Architecture 1.0 Public Review Draft 1 (April 2008)
[17] The Open Group, SOA Working Group SOA Governance Project (August 2006), http://www.opengroup.org/projects/soa-governance/
[18] Arsanjani, A.: Service-oriented modeling and architecture – How to identify, specify, and realize services for your SOA (November 2004), http://www.ibm.com/developerworks/library/ws-soa-design1/
[19] Cherbakov, L., Galambos, G., Harishankar, R., Kalyana, S., Rackham, G.: Impact of service orientation at the business level. IBM Systems Journal 44(4), 653–668 (2005)
[20] Erl, T.: SOA Principles of Service Design, 1st edn. Prentice Hall, Upper Saddle River (2008)
[21] Krafzig, D., Banke, K., Slama, D.: Enterprise SOA: Service-Oriented Architecture Best Practices. Prentice Hall, Upper Saddle River (2006)
[22] Erl, T.: SOA Design Patterns. Prentice-Hall, Upper Saddle River (scheduled for 2008)
[23] Manes, A.T., Tilkov, S.: Anne Thomas Manes on SOA, Governance and REST (July 2007), http://www.infoq.com/
[24] Open SOA Collaboration, Service Component Architecture (SCA) v1.0 (March 2007), http://www.osoa.org/
[25] Melcher, J., Seese, D.: Process Measurement: Insights from Software Measurement on Measuring Process Complexity, Quality and Performance. University of Karlsruhe (TH), Institute AIFB, Tech. Rep. (2008)
[26] Österle, H., Winter, R. (eds.): Business Engineering, 2nd edn. ser. Business Engineering. Springer, Berlin (2003)

Using Enterprise Architecture Models and Bayesian Belief Networks for Failure Impact Analysis

Oliver Holschke[1], Per Närman[2], Waldo Rocha Flores[2], Evelina Eriksson[2], and Marten Schönherr[3]

[1] Technische Universität Berlin, Fachgebiet Systemanalyse und EDV, FR 6-7, Franklinstr. 28-29, 10587 Berlin, Germany
Oliver.Holschke@sysedv.tu-berlin.de
[2] Dpt. of Industrial Information and Control Systems, Royal Institute of Technology (KTH), Stockholm, Sweden
{PerN,WaldoR,EvelinaE}@ics.kth.se
[3] Deutsche Telekom Laboratories, Ernst-Reuter-Platz 7, 10587 Berlin, Germany
Marten.Schoenherr@telekom.de

Abstract. The increasing complexity of enterprise information systems makes it very difficult to prevent local failures from causing ripple effects with serious repercussions to other systems. This paper proposes the use of Enterprise Architecture models coupled with Bayesian Belief Networks to facilitate Failure Impact Analysis. By extending the Enterprise Architecture models with the Bayesian Belief Networks we are able to show not only the architectural components and their interconnections but also the causal influence the availabilities of the architectural elements have on each other. Furthermore, by using the Diagnosis algorithm implemented in the Bayesian Belief Network tool GeNIe, we are able to use the network as a Decision Support System and rank architectural components with their respect to criticality for the functioning of a business process. An example featuring a car rental agency demonstrates the approach.

Keywords: Enterprise Architecture Management, Decision Support Systems, SOA, Bayesian Belief Nets, Diagnosis, Failure Impact Analysis.

1 Introduction

Today's businesses are increasingly dependent upon IT, not only to support their business processes but also to automate their business processes. With the advent of integration technologies the information systems have become more and more interconnected. This means that the decision makers in charge of managing the enterprise information systems have lesser possibilities of knowing how any particular decision concerning changes to the information systems affect other information systems or for that matter the business itself. The process of conducting and planning preventive IT maintenance and allocating maintenance and operation resources where they do the most good is one area where the sheer complexity of the information system poses a problem.

G. Feuerlicht and W. Lamersdorf (Eds.): ICSOC 2008, LNCS 5472, pp. 339–350, 2009.

Enterprise Architecture (EA) is a proposed solution to reduce complexity and allow for better decision-making. Using EA models illustrates the architectural components of the enterprise system and their interconnections in a way that is comprehensible to ordinary people. There is a plethora of EA frameworks currently in use including The Open Group Architecture Framework (TOGAF) [1], the Department of Defense architecture framework (DoDAF) [2], and others [3]. Although many of these frameworks propose models that capture interconnections between systems, they fail to depict causal relations between availability of various architectural elements.

To be able to depict and model causal relations between systems, one might use Bayesian Belief Networks [4], which feature a graphical notation to capture causal relations in a more qualitative fashion. In addition to this, there is also a statistical apparatus behind the graphical notation with which one can quantitatively estimate which decision yields the most benefit. We propose a Decision Support System (DSS) for failure impact analysis for Enterprise Architectures based precisely on the Bayesian Belief Networks (BBN) introduced above. Our proposed management process and underlying DSS address several concerns of enterprise architects identified in the *Enterprise Architecture Management Pattern Catalogue* [5] which has been developed at the Technische Universität München, Germany. Our method can be regarded as an implementation of the *Infrastructure Failure Impact Analysis (M-34)* pattern which addresses concerns about infrastructure failures, including concerns about the probability of these failures being causes for process, service or other element defects. The method of creating the DSS consists of EA models based on the ArchiMate meta-model [6, 7] and their translation to a BBN, then using an algorithm to simulate which architectural element is the most critical. The following section proceeds to describe BBNs in general and diagnostic analysis. Section 3 describes how to apply BBNs and diagnostic analysis and how to create the Decision Support System for failure impact analysis. The creation of the DSS is demonstrated in section 4, applied to an example car rental agency. The diagnostic use of the DSS is illustrated in section 5. Section 6 concludes the paper.

2 Bayesian Belief Networks and Diagnostic Analysis

2.1 Bayesian Belief Networks

A Bayesian Belief Net (BBN) is a graphical model that combines elements of graph and probabilistic theory. A BBN describes a set of causal relations within a set of variables, and a set of conditional independencies including joint probabilities as depicted in Fig. 1. A directed, acyclic graph (DAG) represents the causal dependencies between the variables (or nodes). Each node represents a variable with corresponding conditional probability distribution, displayed in a Conditional Probability Table (CPT). The strength of BBN manifests in the possibility of reasoning about results given certain observations according to Bayesian rules. BBN can answer requests of the form “what, if …” with respect to specific variables. Applied in this way BBN are powerful probabilistic inference machines [8]. Further explanations on the semantics of BBN can be found in e.g., [4, 9].

While the structure of a BBN may in principle be unknown, we propose to exploit the availability of an EA model in which nodes and relations are "known". Doing this relieves us of learning an expressive BBN structure, e.g., by search-and-score procedures [8]. The exact mapping of an EA model to a BBN structure will be explained in section 3 and 4. The parameters of a BBN can either be collected from historical data and/or expert assessments, or learnt via estimation methods such as Maximum-Likelihood or Bayesian Estimation [8, 9]. Fully parameterized BBN can be used for different inferential tasks, i.e., classification (mapping a data element into a previously defined category), prediction (the forecast of a value of a dependent variable given a specific observation), and diagnosis (concluding a possible cause of a changed variable given a specific observation). With respect to the concerns of enterprise architects the use type *diagnosis* is of particular relevance. If the architect – in case of EA changes – had means of identifying causes of disruptive effects, this would benefit his architectural decision-making process in terms of efficiency and effectiveness. For this *diagnostic analysis* can be conducted on a BBN. In the following we briefly describe how diagnosis is applied.

2.2 Diagnostic Analysis

The following description of diagnosis in BBN is based upon [10], a master thesis that specifically describes the implementation of the relevant diagnostic functions in the Bayesian Belief Network modeling tool GeNIe [11] developed at the University of Pittsburgh.

Diagnosis involves two types of tasks: 1) determining the (combination of) causes of the observations, and 2) increasing the credibility of the diagnosis through the collection of additional, initially unobserved, data. Since information seldom comes for free, the second task by necessity involves the formulation of a strategy to gather information as cleverly as possible, i.e., to gain the most diagnostic value at the least cost. We now proceed to make this more precise.

Let a diagnostic probability network (DPN) be defined as a Bayesian Belief Network where at least one random variable H is a hypothesis variable (e.g., an infrastructure element such as a server) and at least one other random variable T is a test variable (those variables of the model that we potentially can collect information about, i.e., a manager's opinion about the availability of an architectural element he is responsible for).

Let $\boldsymbol{H}$ denote the set of all hypothesis variables, and $\boldsymbol{T}$ the set of all test variables. Furthermore, each test $T \in \boldsymbol{T}$ has a cost function $Cost(T)$: $\boldsymbol{T} \rightarrow \boldsymbol{R}$, because usually an effort has to be made to collect data about an object. If a test is free, the associated cost is set to zero. Also, each hypothesis H has an associated value function, $V(P(H))$: $[0,1] \rightarrow \boldsymbol{R}$.

Given a DPN, we have the expected value EV of performing a test $T \in \boldsymbol{T}$:

$$EV(T) = \sum_{t \in T} V(P(H \mid t)) \cdot P(t) \tag{1}$$

To make an informed decision, we also need to account for the expected outcome of not performing the test T. We therefore introduce the expected benefit EB:

$$EB(T) = EV(T) - V(P(H)) = \sum_{t \in T} V(P(H \mid t)) \cdot P(t) - V(P(H)) \tag{2}$$

Still, however, no connection has been made to the cost of the test. This is remedied by the test strength *TS,*

$$TS(H,T) = \frac{EB(T)}{V(P(H))} - K \cdot Cost(T)\,, \tag{3}$$

where we have introduced the coefficient *K*, reflecting the relative importance of the expected benefit versus the cost of the test.

The definition of the value function still remains. To optimize the test selection with respect to multiple hypotheses, a function based on the marginal probability between hypotheses (rather than the joint probability) called Marginal Strength 1 (*MS*1) is introduced [10],

$$MS1(P(F)) = \left(\frac{\sum_{t \in T} (f - 0.5)^2}{0.5^2} - n_F \right) \cdot \frac{1}{n_F}\,, \tag{4}$$

where F is the set of all selected target states f_i of the hypotheses that the user wishes to pursue and n_f is the total number of target states. This test selection function is convex with a minimum at $1 - n_f$ and maxima at 0 and 1. The value function that we are looking for now becomes the sum of the marginal strength for all target states:

$$V(P(F)) = \sum_{f \in F} MS1(P(f))\,. \tag{5}$$

3 Using a BBN for Decision Support in Failure Impact Analysis

3.1 Management Process for Failure Analysis Using a Decision Support System

Before we describe our Decision Support System (DSS), we briefly provide the organizational context of failure impact analysis in Enterprise Architecture. For this we explain a management process that generally needs to be walked through in the case of disruptions in business processes, services or other architectural elements. The management process consists of the following 5 main activities: *1. Observe failure event:* Before any measures can be planned or taken to resolve a failure, the failure event has to be observed e.g., by the responsible enterprise architect. The failure can be an actual observation or be part of a simulation in order to prepare countermeasures for future events. Tools that support the inspection and visualization of failure events are of great assistance to the observing person. The information required to detect failures may be supplied by Business Activity Monitoring (BAM) systems. *2. Set observation in DSS:* The observed failure is provided to the DSS (next section) – either manually through exporting and importing data or automatically in case of an integrated system. *3. Conduct diagnosis*: The DSS conducts a diagnosis

based on the provided observation and delivers a ranked list of architectural elements. The ranking is based on probabilistic information in the DSS and displays what probable causes the observed failure may have. *4. Availability of additional observations:* The person in charge of the failure analysis checks whether additional observations are available (that could also be to check how costly additional observations would be and if that would add more valuable information to the diagnosis). If positive, then this process loops back to step 2 and sets the additional observations in the DSS. If negative, the process can proceed to step 5. *5. Initiate repairing activities according to diagnosis:* Based on the ranked architectural elements resulting from the DSS, repairing activities or projects can be planned and initiated by the responsible architect and/or project manager. The probabilistic ranking shall make the sequential ordering of activities in these projects more efficient and complete.

3.2 Creating the Decision Support System for Failure Analysis Based on BBN

In the following we describe our method of how to create a BBN on the basis of an EA model in order to use it for decision support in EA. Starting point for the construction is a specified EA model. The overall method consists of four main steps. Steps 1 and 2 address the creation of the BBN's structure based on the EA model, i.e., what are the variables and what relations exist between them. Steps 3 and 4 address the parameters of the BBN: during step 3 discrete values for the variables are defined for improved usability; in step 4, the built BBN structure is complemented with conditional probability distributions for all variables. The method and its steps are shown in Fig. 1. The detailed actions within all steps will be explained in the next subsections.

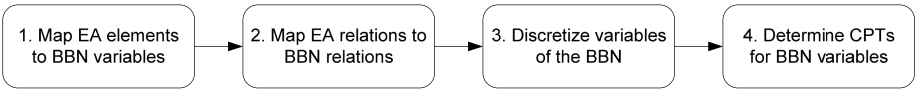


Fig. 1. Method for creating a Bayesian Belief Net on the basis of an Enterprise Architecture model in order to use it as a Decision Support System for failure impact analysis

Mapping the EA Structure to the BBN Structure (Steps 1 & 2)
For obtaining the BBN structure we exploit the fact that in EA models as well as in BBN the central concepts elements and relations are used. Regarding the EA model as a graph allows us to map the EA model to a BBN. We define the following general mapping rule that constitutes step one and two of the method:

1. *Map EA elements to BBN variables:* Each EA element of the EA model maps to a variable (node) in the BBN.
2. *Map EA relations to BBN relations:* Each EA relation between two EA elements in the EA model maps to a causal relationship between two variables (nodes) in the BBN.

BBN as graphs are directed and acyclic. When mapping the EA model to the BBN, directedness and acyclicity must be preserved. Relations that violate this rule either

have to be removed or be modified. For preserving acyclicity see also [12]. The result of the mapping is a DAG consisting of variables and relations representing EA elements and EA relations. Having defined the structure of the BBN, the parameters of all variables now have to be determined.

Discretizing Variables and Determining CPTs (Steps 3 & 4)
Variables in a BBN can, in principle, represent continuous spectra of a specific feature [9]. We focus on discrete states of BBN variables due to successful implementations in other domains [8] and the increased ease for users. Relating this to our EA context, an exemplary discretization of a variable would be: the EA element "Server" has the two discrete states "up" and "down", or, a "Service" has three possible states for response time, i.e., "fast", "moderate" and "slow". Determining the discrete states can be done conjointly with developers and end-users. All these activities can be summed up in step *3. Discretize variables of the BBN*.

Having determined the BBN structure and discretized its variables, the conditional probability distributions for all variables have to be obtained. This constitutes step *4. Determine CPTs for BN variables* of our method. In case of availability of some data, existing mathematical estimation methods can be applied, such as Maximum-Likelihood estimation (MLE) and related estimation algorithms [13]. In addition to this there are many ways to gather empirical data, see for instance [14] or [15].

The collection of data without using any mathematical estimation methods can be done applying one of the following general methods below:

Direct collection (of technical parameters of the actual EA elements; read out log files, etc.);

1. Indirect collection (of data in data bases at distributed locations that allow to draw conclusions about element dependencies);
2. User-based estimation of causal dependencies (by querying the users – via interview or questionnaire).

The manner of data elicitation may also depend on the individual collection strategy of a company. Methods one and two usually require additional technical efforts beforehand because EA elements have to be enabled to provide adequate information to the probabilistic model. Method three does not require these technical efforts. This approach collects relevant conditional probabilities through interviews with e.g., architecture experts, programmers, and system users as well as through analysis of the participating EA elements. On data collection see also [14]. Having collected all conditional probability distributions the BBN is now fully specified and may serve as decision support for failure impact analysis in EA.

4 Scenario-Based Analysis: Creating the Decision Support System

We apply our method to an exemplary Enterprise Architecture to demonstrate the creation of the BBN and its application as decision support for failure impact analysis in EA. The enterprise we chose is a virtual car rental agency with a service-oriented architecture and a real life implementation. The description of the business scenario and the service-oriented architecture and its implementation details can be found in [16]. The core business processes of a car rental agency are 'car reservation', 'car

pick-up', and 'car check-in', i.e., returning the rental car. We analyze the business process of returning a rental car back to the agency, i.e., "Check-in car".

The business process "Check-in car" is initiated at the local service by the return of a car. For this, data about the returned car has to be requested from the system. The car is inspected in presence of the customer and claims are recorded. An invoice is then created automatically based on the rental data and the entered claim information. The monetary quantification of claims and retention is based on a claims list mapping claims to amounts. If there are no claims the car is released right away. If claims are asserted, a claim report is generated. This claim report is submitted to the claim settlement department which is responsible for contacting the insurance company and entering regulation information. At the final stage the case is documented and the car is repaired and released. The corresponding EA is modeled with ArchiMate [6, 7] and is depicted in Fig. 2. In accordance with our method defined in section 4 the following steps are executed to create the BBN.

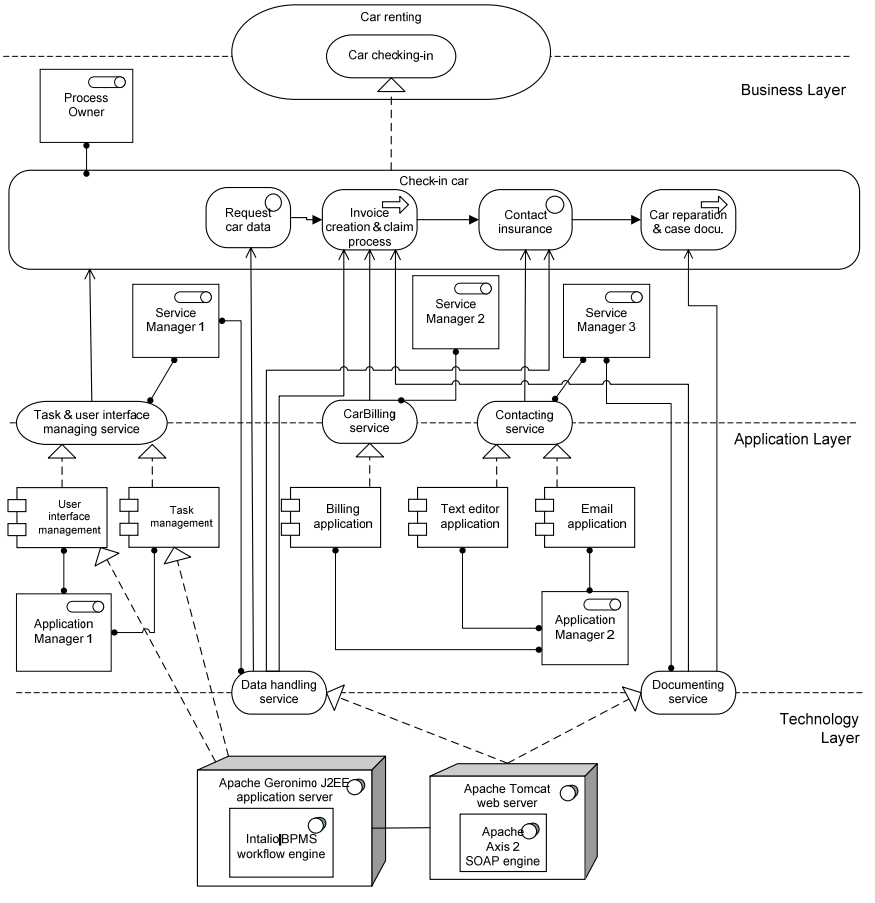


Fig. 2. EA model of the car rental scenario, showing all architectural elements involved in the "Check-in car" process

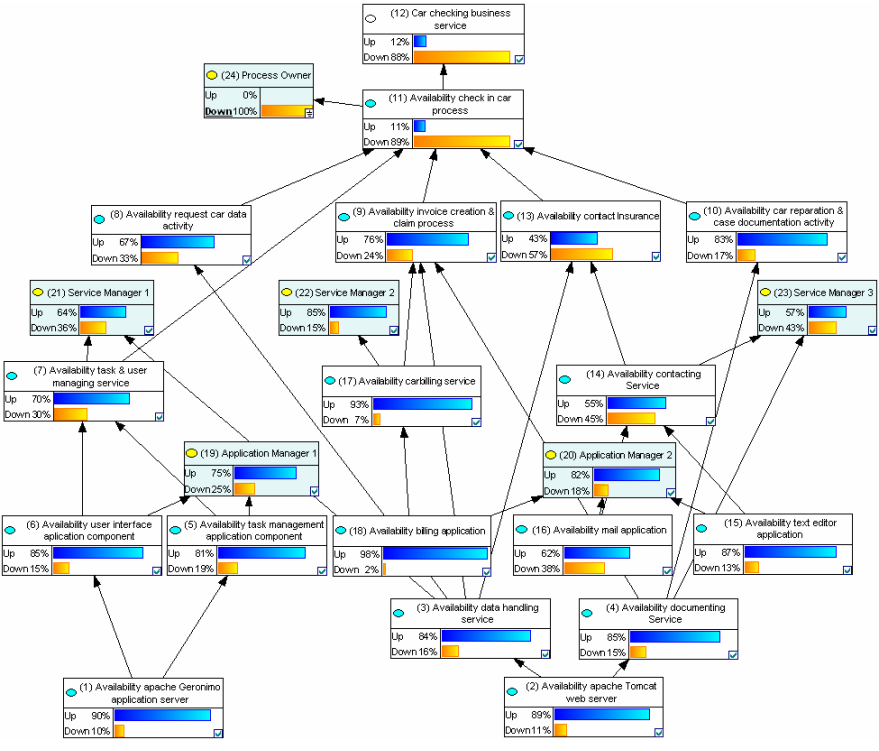


Fig. 3. Structure and parameters of the Bayesian Belief Network mapped from the car rental agency Enterprise Architecture model

Step 1: Map EA Elements to BBN Variables

Out of the 27 architectural elements in the EA model (Fig. 2), 24 elements are mapped to the BBN as variables/nodes. All mapped EA elements are depicted in the BBN in Fig. 3. For instance, the EA element "Apache Geronimo J2EE Application Server" is mapped to node "(1) Apache Geronimo application server", the EA element "Data handling service" between Application and Technology Layer in the EA model is mapped to node "(3) Data handling service" in the BBN, and so on. It has to be noted that not all EA elements have been mapped to the BBN under the assumption that the non-mapped EA elements do not have any causal influence on other elements. This could be because they serve the mere purpose of structuring (e.g., the business service "Car renting") or are on a very deep technological layer, such as the "Intalio|BPMS workflow engine", whose causal relations to the hardware would not contribute to a significant better understanding from a business process management perspective. The latter supports the goal of maintaining a manageable view on the BBN.

Step 2: Map EA Relations to Causal Relationships in the BBN

In the car rental EA model there are mostly *directed* relations. Those relations are mapped to causal relationships in the BBN, e.g., the "Realization" relation (dotted

line, arrowhead not filled) between the "Apache Geronimo J2EE application server" and the application component "Task management" is mapped to a causal relationship between node "(1) Availability Apache Geronimo application server" and node "(5) Availability Task management application component" in the BBN (as in Fig. 3).

There are also eleven *undirected* relations, i.e., those relations between business roles and business processes/services/applications (e.g., between "Process Owner" and the "Check-in car" business process). We say that the people who adopt a specific business role are able to assess the status of the architectural element they are responsible for to a certain extent, e.g., a process owner can make a judgment on the availability of a business process. This observation is based on the actual status of the element. Therefore we can map the undirected EA relations to directed causal relationships going from the element to the observer, indicating that the observation of an element by a person will usually be influenced by the actual status of the element. Having mapped the undirected relations to directed ones, we fulfill the criterion of directedness required by BBN.

Due to the strictly maintained paradigm of service-oriented architecture in the scenario, any cycles in the EA model are absent. Thus, the step in our method which removes any cycles from the BBN is not required here (for cycle removal see [12]). As opposed to the traditional top-down build-up of BBN, we model the causal relationships and nodes upwards – like a bottom-up growing tree – to maintain the resemblance with the EA model.

Step 3: Discretize Variables of the BBN

Each EA element that we have identified as a BBN node could be described by various features. Depending on the value of a feature of one variable, which could in principle stem from a continuous spectrum, the feature(s) of other variables are influenced. An important feature of Enterprise Architecture elements that we focus on is the *availability* of these elements [17]. We therefore apply the feature „Availability" and define two discrete, mutually exclusive values: "up" and "down" (see also [8, 12]). We discretize the value spectrum for our single-feature variables to keep the complexity of the network nodes manageable. Moreover, these two values have shown to be generally comprehensible states of different system elements during talks with system experts and users. Also, for a user it will be generally easier to give estimations on the conditional probability distribution of *two* states of an element, rather than to estimate distributions between three or even more states. The EA model-based BBN structure is depicted in Fig. 3.

Step 4: Determine CPTs of the BBN variables

In this example we have not engaged in the elicitation of data to set the CPTs according to the methods proposed by for instance [13], [15] or [14]. To simulate the actual approach we randomly assigned numbers to the CPTs for the BBN in Fig. 3 above.

5 Using the Decision Support System for Failure Impact Analysis

We use the DSS as we have described in the management process in section 3.1 according to diagnostic analysis (section 2.2) to localize probable causes of an

observed failure. To demonstrate the usage of the DSS we let the "Process Owner" observe that the business process "Check-in car" is *down*. This relates to the first step in the management process: *Observe failure event*. The next step is to set the observation in the DSS, i.e., the modeled Bayesian Belief Network, according to the observation. This is done in the GeNIe-tool by *setting the evidence* of node "(24) Process Owner" to "down" (see Fig. 3). During the third step, *Conduct Diagnosis*, the actual diagnostic analysis based on the one observation and the BBN model is executed (in GeNIe, the *Test Diagnosis* button initiates this). In our example the cost of observing the availability status of architectural elements is still set to zero, assuming that querying service or application managers is an effortless task. This means that additional observations will always contribute to a better diagnosis since the observation is for free. For a more realistic representation in the future the costs of asking people or having people to publish their observations need to be introduced in the model.

After the observation, diagnostic analysis calculates the a-posteriori probabilities of all target nodes. In Fig. 4, left (screenshot taken from GeNIe [11]), the diagnostic results are depicted as a list of ranked (ranked according to the probability of being *down*) target nodes. The ranked target node list starts with nodes (11): 0.892, (13): 0.566, (14): 0.447… and so forth. This information points an enterprise architect to architectural elements that ought to be attended to immediately – those having high probabilities of unavailability and being the possible cause of the process failure – compared to those elements with lower probabilities. Based on the given information this would be the best order of activities in a repair project.

The order of activities can change when more information of the status of architectural elements is known. In addition to the formerly observed business process "Check-in car" being *down*, we let "Service Manager 3" observe the services "Car documenting service" and "Contacting service" under his responsibility being available, i.e., *up* (Fig. 4, right). The ranking of target nodes significantly changes, after having this additional observation. For instance, node *(14) Availability contacting service: down* had a probability of 0.447. Taking into account the new observation, the probability of node (14) being *down* drops to 0.175. This provides useful information to the enterprise architect who can now initiate differently ordered activities to repair the failure.

Ranked Targets	Probability
(11) Availability check in car process:Down	0.892
(13) Availability contact Insurance:Down	0.566
(14) Availability contacting Service:Down	0.447
(16) Availability mail application:Down	0.381
(8) Availability request car data activity:Down	0.332
(7) Availability task & user managing service:Down	0.302
(9) Availability invoice creation & claim process:Down	0.243
(5) Availability task management application component:Down	0.187
(10) Availability car reparation & case documentation activity:Down	0.167
(3) Availability data handling service:Down	0.164
(4) Availability documenting Service:Down	0.154
(6) Availability user interface aplication component:Down	0.153
(15) Availability text editor application:Down	0.135
(2) Availability apache Tomcat web server:Down	0.112
(1) Availability apache Geronimo application server:Down	0.097
(17) Availability carbilling service:Down	0.069
(18) Availability billing application:Down	0.020

Ranked Targets	Probability
(11) Availability check in car process:Down	0.847
(13) Availability contact Insurance:Down	0.387
(7) Availability task & user managing service:Down	0.356
(8) Availability request car data activity:Down	0.325
(16) Availability mail application:Down	0.263
(5) Availability task management application component:Down	0.218
(6) Availability user interface aplication component:Down	0.178
(9) Availability invoice creation & claim process:Down	0.175
(14) Availability contacting Service:Down	0.175
(3) Availability data handling service:Down	0.117
(1) Availability apache Geronimo application server:Down	0.114
(10) Availability car reparation & case documentation activity:Down	0.089
(15) Availability text editor application:Down	0.084
(17) Availability carbilling service:Down	0.069
(4) Availability documenting Service:Down	0.046
(2) Availability apache Tomcat web server:Down	0.043
(18) Availability billing application:Down	0.020

Fig. 4. Left: Diagnostic results as a list of ranked target nodes after one observation, and Right: Additional observation: differently ordered ranked target nodes after two observations

6 Conclusion

We have proposed a DSS based on a Bayesian Belief Network to address one important concern of today's Enterprise Architects, i.e., conducting failure impact analysis and diagnosis in EAs. The BBN was created based upon an architectural model of an Enterprise Architecture, i.e., the knowledge about causal dependencies between actual architectural elements was exploited to create the BBN nodes and relationships to capture the uncertainties. Particularly for architectures with growing complexity, those approaches which capture this uncertain knowledge and still allow reasoning on it seem suitable. This is supported by situations in which the actual states of system elements cannot be determined, but only *observed* by managers using there experience.

We have designed a detailed method to create the Bayesian Belief Network-based DSS consisting of the mapping of EA model elements to a BBN and eliciting all its required structural features and probabilistic parameters. To demonstrate the general feasibility of the approach we created a DSS for an EA of a car rental agency and conducted an exemplary failure diagnosis in it, giving managers valuable information about what elements could be the probable causes. Even though our exemplary scenario is a complete service-oriented Enterprise Architecture, more complex structures of EA models (e.g., those including loops, non- and bi-directed relations) in other enterprises are definitely possible. In these cases additional mapping rules – from the EA modeling language to BBN parameters – need to be introduced in order to remove irregularities and create a formally correct Bayesian Belief Network.

In further work we will concentrate on the elicitation of probabilities to populate the CPTs of the Bayesian Belief Network. The spectrum of possibilities, e.g., leveraging expert opinions on system element behavior in contrast to exploring automatic ways of using monitoring information about system availability and other qualities, needs to be analyzed considering the trade-off between expressiveness and cost of elicitation.

References

1. The Open Group: The Open Group Architecture Framework (TOGAF), version 8 Enterprise Edition. The Open Group (2005)
2. Department of Defense Architecture Framework Working Group: DoD Architecture Framework Version 1.0 Department of Defense, USA (2004)
3. Schekkerman, J.: How to survive in the jungle of Enterprise Architecture Frameworks. Trafford, Victoria, Canada (2004)
4. Friedman, N., Linial, M., Nachman, I.: Using Bayesian Networks to Analyze Expression Data. Journal of Computational Biology 7, 601–620 (2000)
5. Buckl, S., Ernst, A.M., Lankes, J., Matthes, F.: Enterprise Architecture Management Pattern Catalogue, Version 1.0. Technische Universität München, München (2008)
6. Buuren, R.v., Hoppenbrouwers, S., Jonkers, H., Lankhorst, M., Zanten, G.V.v.: Architecture Language Reference Manual. Telematica Instituut, Radboud Universiteit Nijmegen (2006)

7. Jonkers, H., Lankhorst, M.M., Buuren, R.v., Hoppenbrouwers, S., Bonsangue, M.M., Torre, L.W.N.v.d.: Concepts For Modeling Enterprise Architectures. Int. J. Cooperative Inf. Syst. 13, 257–287 (2004)
8. Lauría, E.J.M., Duchessi, P.: A Bayesian Belief Network for IT implementation decision support. Decision Support Systems 42, 1573–1588 (2006)
9. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern classification. Wiley Interscience, Hoboken (2000)
10. Jagt, R.M.: Support for Multiple Cause Diagnosis with Bayesian Networks. Vol. M. Sc. Delft University of Technology, the Netherlands and Information Sciences Department, University of Pittsburgh, PA, USA, Pittsburgh (2002)
11. Decision Systems Laboratory: GeNIe (Graphical Network Interface). vol. 2008, University of Pittsburgh (2008)
12. Tang, A., Nicholson, A.E., Jin, Y., Han, J.: Using Bayesian belief networks for change impact analysis in architecture design. Journal of Systems and Software 80, 127–148 (2007)
13. Neapolitan, R.: Learning Bayesian networks. Prentice-Hall, Inc., Upper Saddle River (2003)
14. Keeney, R.L., Winterfeldt, D.v.: Eliciting Probabilities from Experts in Complex Technical Problems. IEEE Transactions On Engineering Management 38 (1991)
15. Woodberry, O., Nicholson, A.E., Korb, K.B., Pollino, C.: Parameterising Bayesian Networks Australian Conference on Artificial Intelligence. Springer, Heidelberg (2004)
16. Holschke, O., Gelpke, P., Offermann, P., Schröpfer, C.: Business Process Improvement by Applying Reference Process Models in SOA - a Scenario-based Analysis. Multikonferenz Wirtschaftsinformatik. GITO-Verlag, Berlin, München, Germany (2008)
17. International Standardization Organization and the International Electrotechnical Committee: ISO/IEC 13236 - Information technology — Quality of service: Framework. ISO/IEC (1998)

Assessing System Availability Using an Enterprise Architecture Analysis Approach

Jakob Raderius, Per Närman, and Mathias Ekstedt

Department of Industrial Information and Control Systems,
Royal Institute of Technology (KTH), Stockholm, Sweden
raderius@kth.se, {pern,mek101}@ics.kth.se

Abstract. During the last decade, a model based technique known as enterprise architecture has grown into an established approach for management of information systems in organizations. The use of enterprise architecture primarily promotes good decision-making and communication between business and IT stakeholders. This paper visualizes a scenario where enterprise architecture models are utilized to evaluate the availability of an information system. The problem is approached by creating models based on metamodels tailored to the domain of enterprise architecture analysis. As the instantiated models are fed with data particular to the information system, one can deduce how the organization needs to act in order to improve the system´s availability.

Keywords: Enterprise architecture, architecture analysis, Bayesian networks, decision graphs, availability.

1 Introduction

The discipline of Enterprise Architecture (EA) advocates using models to capture IT-systems, business processes and their interrelations. The purpose of these models is to visualize the business and IT in a way that abstracts from the complexity inherent in modern enterprise information systems thus facilitating IT management and ensuring business-IT alignment [1].

When faced with architectural changes, decision-makers can use EA models from an enterprise analysis framework for decision-support. To facilitate decision-making, the analysis framework must preserve and explicitly display the uncertainty of the analysis. Uncertainty is frequently abundant when performing architecture analysis. To remedy this, a formalism with which one is able to perform quantitative architecture analysis with information which may be uncertain, such as interview answers, was presented in [4]. This formalism is called Extended Influence Diagrams (EID).

In addition to the ability of capturing the uncertainty of the world, the EIDs are able to model causal properties of the world. This entails that the result of the architecture analysis will not only show which decision alternative is better with respect to the goal at hand, but also why this is so, i.e. which factors make one alternative better than its competitors.

G. Feuerlicht and W. Lamersdorf (Eds.): ICSOC 2008, LNCS 5472, pp. 351–362, 2009.

Using an EID for architecture analysis imposes information requirements on the architecture models, i.e. the models and in particular their underlying metamodels must answer the questions posed by the EID.

This paper describes an EID for availability assessment, how it was used together with associated metamodels for analysis of a set of system properties according to the ISO 9126 standard [6][7]. Specifically, the paper describes the application of the approach in a case study of a specific system at a Swedish company. The system in question is an enterprise data warehouse whose initial goal was to create an infrastructure not limited to current functional requirements of data, but could accommodate future changes in information structure due to a generic and extensible nature. Although the case study analyzed the system properties maintainability, availability; interoperability and functional suitability space limitations only permits us treating the availability assessment. For a full account readers are referred to [22].

The remainder of this paper consists of six sections. Next is a related works chapter which treats other approaches to IT system availability assessment. Section 3 gives an overview of the analysis framework, and its two main components; EIDs and metamodels. Section 4 is dedicated to the concept of quality of service and the quality attributes. Section 5 describes the case study and Sections 6 and 7 present the analysis and conclusion respectively.

2 Related Works

There are many methods for availability analysis. Commonly used methods to estimate availability include reliability block diagrams and Monte Carlo simulations [23]. All of these approaches are unable to i) express and manage input data uncertainty, and ii) rely heavily on the building of intricate models of, primarily, hardware architectures. In this case study, decision-makers had to rely on vague and incomplete information collected mainly through interviews – due to the lack of proper documentation - and it was therefore crucial to be able to express the level of certainty of the analysis.

As for ii), the scope of the present study made the creation of elaborate models of the system's architecture prohibitively expensive. Especially since detailed topologies of networks, nodes and other infrastructure documentation was more or less unavailable.

Due to i) and ii), the approaches to availability analysis as presented in [23] and as further elaborated in [24], [25] and [26] were not applicable in this case. Queuing models [27] and Petri nets [23] also require exact and complete data and must be discarded for the same reasons.

Further, iii), it was hinted by the stakeholders that the problems concerning the availability of the system are at least partly a result of problems within the organization operating it, and not simply the result of an inferior implementation.

The analysis approaches mentioned above are unable to perform any kind of organizational analysis, and are therefore insufficient with respect to iii).

3 A Quality of Service Evaluation Framework

This chapter will present the skeleton of a framework which can be used to create models that will answer a very specific set of questions about the quality of service of

enterprise information systems. The approach utilizes a model-based approach to achieve a level of abstraction sufficient to perform analyses concerning system qualities such as the functionality, availability, usability, performance, security and interoperability.

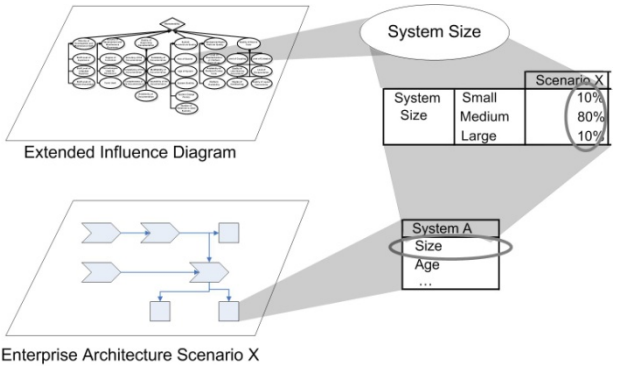


Fig. 1. To support architecture analysis, the architecture models must contain information specified by the analysis framework, here captured in the form of an extended influence diagram

Models are created with a specific goal in mind, such as increased system availability, and allow reasoning about the consequences of various scenarios with various modifications of variables. To guarantee that the models support system quality analyses, the underlying metamodel must contain at least the information required by the analysis frameworks for the respective system qualities. For example, if the goal is to analyze system maintainability the analysis framework might postulate that the analyst must determine the size of the system. If the analyst uses a model-based approach this means that the model must contain the property "system size". See Figure 1. Since the content of the architecture model is determined by their underlying metamodel, the model's metamodel must be closely coupled with the analysis frameworks.

In order to break down system characteristics into concrete and measurable questions suitable for analysis, we capture the characteristics using Extended Influence Diagrams (EID). These are graphic representations of decision problems and can be used to formally specify enterprise architecture analyses [4][7]. EIDs are an extension of influence diagrams, [18] [19] and feature the probabilistic inference engine taken from Bayesian networks [5][20], with which it is possible to reason about causal relations between different nodes through the means of Conditional Probability Tables.

Nodes of EIDs could in this context for example be "system coupling", or "system maintainability", and capture relations of the real world, such as "lower system coupling increases the system maintainability" through casual relationships. EIDs allow each node [7] to be broken down into sub-nodes until a sufficient level of tangibility adequate for data collection and analysis is achieved. To automate the

rather cumbersome probabilistic calculations of Bayesian networks there are several software tools available; see for instance GeNIe [21] which was used in this study.

Even though EIDs formally specify characteristics that influence a system and the causalities that exist between them they are in themselves insufficient to perform architecture analysis. To be able to create and measure different scenarios where those characteristics assume different states one needs a model better suited for analysis and use-case-specific instantiation. Class diagrams capture the entities of an architecture and their entity-relations and thus complement the EIDs that merely describe entity properties and their causal relations. The concept of *abstract models* [28] extends regular class diagram metamodels by layering an extended influence diagram on the metamodel. The nodes of the EIDs are represented as attributes of the metamodel and the causal relations between the attributes as *attribute relations*.

An abstract model thus contains the entities and attributes necessary to conduct analyses of a certain system. The entities can be extracted from EIDs and given attributes and relations based upon the stated causal relations [7].

3.1 Using the Framework

The analysis frameworks capture theories concerning architecture analysis. This theory is expressed in the abstract models comprising an EID and a metamodel. When using the approach, appropriate data is collected to create architecture models. The data collected is used to set the states of the attributes of the model entities. This information can be used as input to the Bayesian inference engine underlying the EIDs which makes the task of analysis straightforward. Values are propagated throughout the structure of the EIDs, and yield a quantitative result.

4 Quality of Service

The term quality of service resembles and is based on software quality as defined in the first part of the ISO-specification 9126 - 1 [6]. Its software quality metrics roughly correspond to the quality attributes of quality of service. The metrics found in ISO 9126 are, however, too low level, requiring massive amounts of data which make them inapplicable for the kind of high-level analyses that are performed here.

Quality of Service is defined in terms of four are stand-alone main quality attributes and four sub-attributes of the fifth main attribute; functionality. The **availability** of an information system is a measure of how often it is ready to deliver its services to its users [8]. The **usability** is a measure of how easy it is for a user to interact with and perform his or her tasks in the system [9]. The **efficiency** represents the degree to which a software system can meet its objective in terms of scalability and responsiveness [10]. Further**,** the **maintainability** represents the ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment [11]. The **accuracy** is one of four sub-attributes that comprise functionality, and represents the degree to which a system, given correct input data, produces output data that is both

accurate and precise [12]. The **security** – another functionality sub-attribute – is the system´s ability to maintain data confidentiality, integrity and availability [13].The **interoperability** represents the ability of two or more systems or components to exchange information and to use that information [11]. Last of the functionality sub-attributes, the **suitability** of an information system is the degree to which the functionality of the system supports its intended work tasks (i.e. the match between the system and its functional requirements) [14][15]. We now proceed to describe the breakdown of the attribute availability [22], where each major area influencing the availability of an information system is given its separate sub-heading. See Figure 2 below for an overview EID for availability assessment.

4.1 Quality of the Maintenance Organization

The **quality of the maintenance organization** is crucial to the availability of a system and this quality is dependent, among other things, on the maintenance organization´s **service level management** – the understanding, enforcement management and monitoring of the expected requirements and service levels [16].

In order to successfully manage these requirements and to enable a system-evolution that continuously upholds them, they have to be clearly understood. Only then can an enterprise information system be managed purposefully as part of a larger concept [17].

Since it is in most systems´ nature to evolve, **change management** is of utter importance, dependent first and foremost upon a change management process and its grade of use. This process includes change requests, as well as approval processes for these requests to ensure that sufficient levels of financial control are exerted [17].

Problem management - incident and problem detection and handling – is, of course, fundamental to correct operation and high uptime. Availability of appropriate resources is a must, as well as quick responsiveness of those resources.

4.2 Reliability

Reliability is the ability to perform under stated conditions for a stated period of time. Dependencies consist of redundancy, the quality of fault prediction and avoidance, and the reliability of sub-components.

Redundancy is essentially a mix of different fail-over solutions. The extension to which software redundancy solutions such as separate memory management and process isolation have been deployed plays a significant part, as well as hardware solutions such as backup disks, networks or complete fail-over servers [16].

When it comes to the **quality of fault prediction and avoidance** in the case study-system, a special subsystem is responsible for active and pro-active defense against faulty and/or incomplete data as well as quality assurance of incoming data deliveries. It does this by analyzing and benchmarking data against validation rules and descriptions of each data supplier´s format and alerting personnel when errors are encountered, even sending data for manual correction when the need arises.

Most systems are divided into **sub-components**, containing data tiers or runtime code, for example. Those are mostly interdependent in the sense that the whole system will not be working correctly without all of them.

For example, although the data receiving component of the system will still be operational if the validation component stops working, the system seen from a customer point of view will not be able to deliver the agreed service levels (in this case, possibly inferior quality data). Thus, the system´s reliability depends strongly upon the reliability of its sub-components [16].

4.3 Recoverability

The **recoverability** of a system is its ability to bypass and recover from a failure [16]. It is dependent upon automatic mechanisms that can be sprung into action as soon as something bad happens (failure with data loss etc.), as well as mechanisms that pro-actively monitor the system and generate the appropriate warning and/or take the appropriate action as soon as possible. Thus, the **quality of automatic fault recognition** and the **fastness of automatic recovery techniques** play important roles.

If failure occurs and leads to corrupt data, speed is of the essence. While nightly jobs creating backups and scripts to automatically restore data when necessary are all valid and fairly standard recovery techniques in the realm of enterprise data warehousing, it is the **fastness** of the initiation of the rollback of transactions gone wrong or the restoration of corrupted databases that matter the most.

4.4 Serviceability and Manageability

Serviceability is defined as the ability to perform effective problem determination, diagnosis and repair [16]. It is dependent upon the **quality of the system documentation** [16][17]. The serviceability of a system is highly dependent upon concise, up-to-date documentation, especially since the maintenance organization changes, and with it the personnel responsible for risen issues. When breaking down the quality of the system documentation into its constituents, one will find dependencies such as the availability, completeness and accuracy of said documentation, among other things.

No less important is the ability of the maintenance personnel to be able to understand what has gone wrong and where. **Understandability of error messages** is of paramount importance, as no incident management can take place if the nature of the incident is unknown [16].

Finally, **manageability**; the ability to create and maintain control over an environment in which the system works correctly, has a highly important dependability upon correct specifications on how the system works and behaves [16]. The continuous management of a system will no doubt see it evolve, and the organization has to ensure that it evolves in a controlled fashion.

An EID for availability analysis containing all the relevant aspects as listed above is shown in Figure 2.

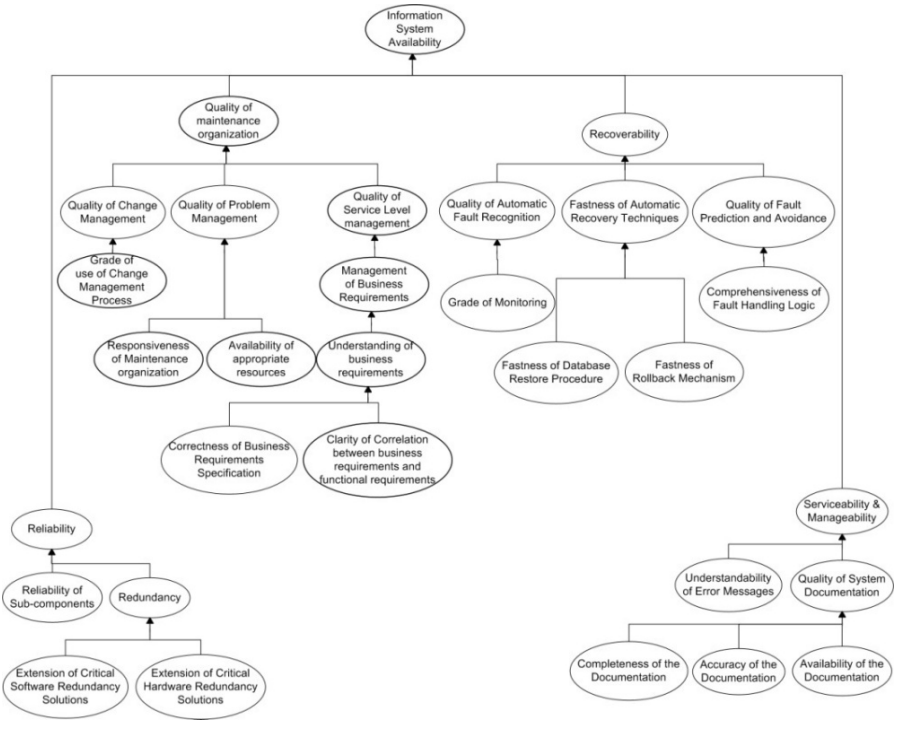


Fig. 2. An availability Extended Influence Diagram

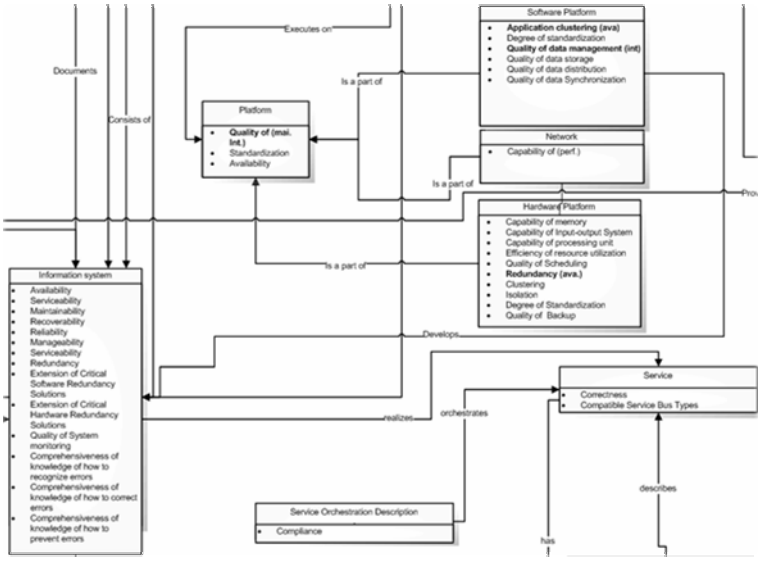


Fig. 3. Part of the metamodel for availability assessment

4.5 A Metamodel for Availability Assessments

A metamodel was derived based on the EID, the metamodel is very similar to the one presented in [7]. This metamodel is generic, containing classes such as **information system** and **hardware platform**. A small part of the metamodel for the case study system is shown in Figure 3. The metamodel can be found in its entirety in [22].

5 Case Study – Evaluating the Availability of a Data Warehouse

In order to analyze the availability of the enterprise data warehouse, data from the system and the organization was collected. Sources include documentation, logged statistics and interviewees. Questions are based on the theory captured in the EID presented in the previous section and the frequencies of the answers were mapped to discrete states to be useful for the Bayesian inference engine of the EID.

Figure 4 depicts a scenario where interviewees with varying degree of certainty and credibility (see [29] for an account of how to estimate source credibility) are asked questions related to change management, to facilitate an estimation of the grade of use of the change management process.

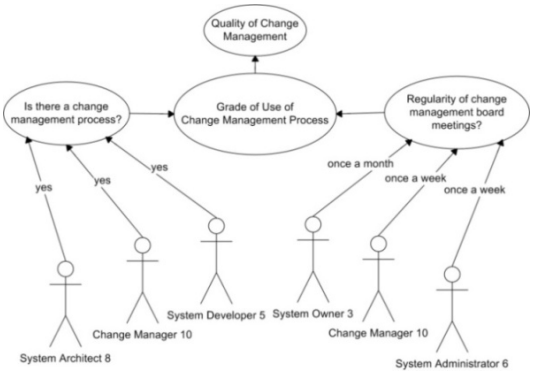


Fig. 4. Collecting data through interviews

As this data, shaped from answer frequencies and aptly modified for their source's credibility is then used in the Bayesian network tool GeNIe to perform the analysis. Collecting data using statistics or documentation is done in a similar fashion.

5.1 Instantiating the Models

By instantiating the metamodel into models it is possible to establish a detailed view of the system, its environment and the links between different parts of the platform and functions within the organization.

With such a model, one is able to reason clearly about how different entities in the system are coupled, and gather insight into how changes in variables will impact the system and the organization. Classes are instantiated using a notation similar to the

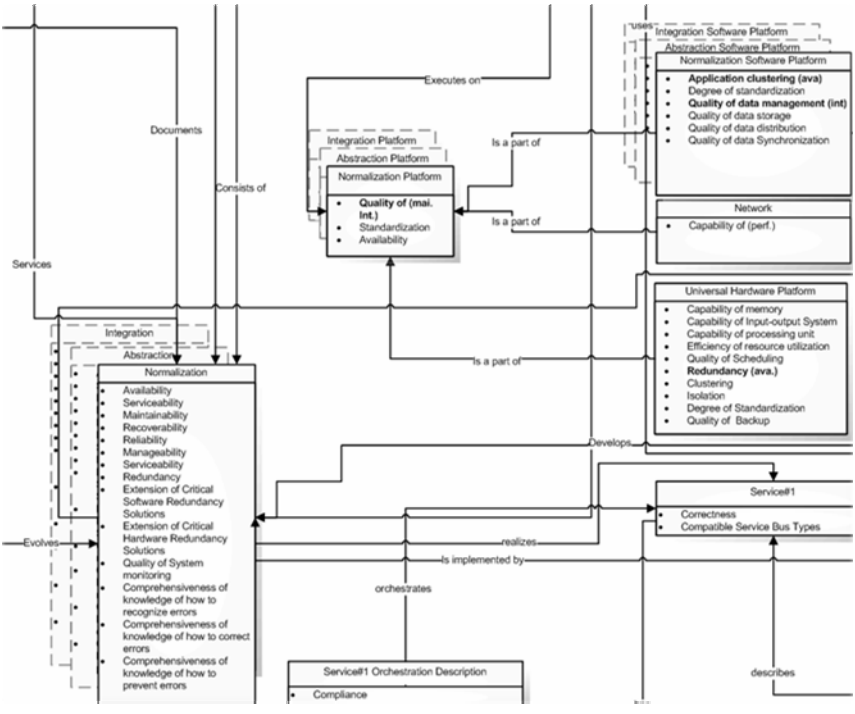


Fig. 5. Part of the model based on the availability metamodel

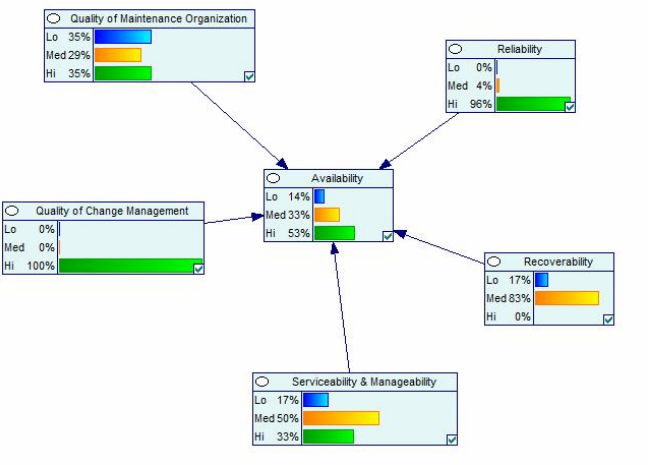


Fig. 6. Availability results

one used for the generic metamodel, though all instances except the top one use a crosshatched line to mark the border of the box representing it, see Figure 5 for an excerpt of the model of the evaluated system.

The instantiated model contains class instances for the three **layers** of the data warehouse, as well as separate instance for the **platforms** and **software platforms** used.

5.2 Results

Figure 6 shows the results for the availability assessment of the data warehousing solution. It should be interpreted as the probabilities that availability reaches a certain state. In this case, it is high with 53%, medium with 33% and low with 14% probability. As the intergroup weights of the different influences for this particular system are not known, weighted sampling distribution is used to distribute the proportionality of the influences in the GeNIe model evenly.

6 Analysis

According to the results displayed in Figure 6, one can say with 53% certainty that the availability of the case study system is high. High in this context translates to the 99.8-99.9% uptime necessary to uphold the organizations´ established service level agreements with customers. Thus, this quality attribute could certainly use some improvement.

The availability causalities found in the EID for availability (Figure 2) state that the quality of the maintenance organization, the Recoverability and the Serviceability & Manageability of the system are main influences and the GeNIe model states that they are 35%, 17% and 17% low respectively.

In the case of the maintenance organization one can tell from the GeNIe model that the low states originate in the lack of a business requirements specification and clarity of the correlation between business requirements and system functional requirements. Hence, a clear link between business concerns and IT should improve availability.

The recoverability is another source of concern, mainly because of a low quality of the fault prediction and avoidance system. This in turn is caused by a very limited set of fault handling logic implemented.

Suggestions for improvement include developing a monitoring and message handling application to be able to respond to incidents more effectively and efficiently, and implementing a sub-system that monitors the system and is able to handle software fatalities by running different command scripts. When it comes to Serviceability & Manageability, efforts to improve system documentation would greatly improve availability.

7 Conclusion

This paper has presented a case study where availability was assessed using an extended influence diagram – a form of Bayesian network - combined with an enterprise architecture metamodel. The metamodel makes it possible to create models with precisely the information that is needed for quantitative system quality analysis.

Furthermore, the paper discussed how a model for availability can be built from a theoretical foundation and then injected with empirical data in order to find the actual

availability of a data warehouse solution. The analysis provided suggestions for how improvement of certain aspects of the data warehouse and its surrounding organization can improve the system's availability.ï

References

1. Minoli, D.: Enterprise Architecture A to Z. s.l.: Auerbach (2008)
2. Zachman, J.A.: Concepts of the Framework for Enterprise Architecture (1992)
3. The Open Group Architecture Framework. TOGAF 8 Enterprise Edition. The Open Group, http://www.opengroup.org/togaf/
4. Johnson, P., et al.: Enterprise Architecture Analysis with Extended Influence Diagrams. Information System Frontiers, vol. 9. Springer, Netherlands (2007)
5. Jensen, F.: Bayesian Networks and Decision Graphs. Springer, New York (2001)
6. ISO/IEC. 9126-1 Software Engineering - Product Quality - Quality Model (2001)
7. Närman, P., Johnson, P., Nordström, L.: Enterprise Architecture: A Framework Supporting System Quality Analysis. In: Proceedings of the 11th International EDOC Conference (2007)
8. Hawkins, M., Piedad, F.: High Availability: Design,Techniques and Processes. Prentice Hall, Upper Saddle River (2001)
9. Nielsen, J.: Usability Engineering. Academic Press, San Diego (1993)
10. Smith, C.U., Williams, L.G.: Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software. Pearson Education, Indianapolis (2001)
11. IEEE. IEEE Standard Glossary of Software Engineering Terminology (1990)
12. Redman, T.: Data Quality for the Information Age. Artech House, Norwood (1996)
13. Stoneburner, G.: Underlying Technical Models for Information Technology Security. National Institute of Standards and Technology, Gaithersburg (2001)
14. ISO/IEC. 9126-2 Technical Report - Software Engineering – Product Quality - Part 2: External Metrics (2003)
15. Sommeville, I., Sawyer, P.: Requirements Engineering. Wiley, Chichester (2004)
16. Johnson, P., Ekstedt, M.: Enterprise Architecture - Models and Analyses for Information Systems Decision Making. s.l.: Studentlitteratur (2007) ISBN 9789144027524
17. Addy, R.: Effective Service Management - To ITIL and Beyond. s.l. Springer, Heidelberg (2007)
18. Shachter, R.: Evaluating influence diagrams. Operations Research, vol. 34(6). Institute for Operations Research and the Management Sciences, Hanover Maryland (1986)
19. Howard, R.A., Matheson, J.E.: Decision Analysis. Influence Diagrams, vol. 2(3). Institute for Operations Research and the Management Sciences, Hanover Maryland (2005)
20. Neapolitan, R.: Learning Bayesian Networks. Prentice-Hall, Inc., Upper Saddle River (2003)
21. GeNIe & SMILE. GeNIe Website (2008), http://genie.sis.pitt.edu
22. Raderius, J.: Assessing the quality of service of an enterprise data warehouse. ICS, KTH, Stockholm (2008)
23. Trivedi, K., et al.: Achieving and Assuring High Availability. LNCS. Springer, Heidelberg (2008)
24. Sahner, R.A., Trivedi, K.S., Puliafito, A.: Performance and Reliability Analysis of Computer Systems. Kluwer Academic Press, Dordrecht (1996)
25. Trivedi, K.S.: Probability & Statistics with Reliability, Queueing and Computer Science Applications, 2nd edn. John Wiley, New York (2001)

26. Zhou, L., Held, M., Sennhauser, U.: Connection availability analysis of span-restorable mesh networks. Springer Science, Heidelberg (2006)
27. Grønbæk, J., et al.: Client-Centric Performance Analysis of a High-Availability Cluster (2007)
28. Johnson, P., et al.: A Tool for Enterprise Architecture Analysis. In: Proceedings of the 11th International EDOC Conference (2007)
29. Gammelgård, et al.: Architecture Scenario Analysis – Estimating the Credibility of the Results. In: Proceedings of the Seventeenth Annual International Symposium of The International Council on Systems Engineering (2007)

An Information Model for Landscape Management – Discussing Temporality Aspects

Sabine Buckl, Alexander Ernst, Florian Matthes, and Christian M. Schweda

Technische Universität München, Institute for Informatics,
Boltzmannstr. 3, 85748 Garching, Germany
{buckls,ernst,matthes,schweda}@in.tum.de
http://www.systemcartography.info

Abstract. Planning, managing, and maintaining the evolution of the application landscape is a focal point of enterprise architecture (EA) management. Whereas, planning the evolution of business support provided by the business applications is understood as one challenge to be addressed in landscape management, another challenge arises in the context of traceability of management decisions.

This paper discusses the requirements regarding support for landscape management as risen by practitioners from industry, gathered in an extensive survey during which the tool support for EA management was analyzed. Thereby, a lack of support for this management discipline was discovered, which is caused by the way, application landscapes are modeled in tools. We subsequently discuss how to incorporate these requirements into an information model.

Keywords: EA management, modeling, temporality, historization.

1 Motivation and Introduction

Over the last years enterprise architecture (EA) management has become an important management area, many companies are currently executing or planning to introduce in the nearby future. As a consequence, a multitude of methods for EA management has been developed by academic communities (cf. [1,2,3]), standardization bodies (cf. [4]), or practitioners (cf. [5,13]). Although these methods differ substantially concerning the quantity, abstractness, and granularity of the EA documentation, needed for EA management, the need for a documentation is common. Thus, different methods and models for creating such a documentation as well as for maintaining its timeliness have been subjected to research, commonly attributing this documentation as a model of the EA (cf. [6]).

The methods and models developed have to address different challenges arising in the context of EA management, especially when the management of the application landscape[1] is concerned. During information gathering not only information about the as-is situation of the landscape has to be collected, but also

[1] The term *application landscape* refers to the entirety of the business applications and their relationships to other elements, e.g. business processes in a company. We do not use the term *application portfolio*, which we regard to have a narrower focus.

G. Feuerlicht and W. Lamersdorf (Eds.): ICSOC 2008, LNCS 5472, pp. 363–374, 2009.

information about future aspects, e.g. projects changing the application landscape, has to be maintained. In order to get an overview on the relationships and dependencies of the various elements of the enterprise, different kinds of visualizations, which we refer to as *software maps*, are typically used (see e.g. V-30 in [1]). Different versions of visualizations are commonly used to illustrate the evolution of the application landscape, either the status quo or future business support. In order to create these documentations, the respective data has to be stored in a repository corresponding to an information model, which defines the elements and the moment in time the information is related to (*planned for*).

Furthermore, landscape management is closely connected to project portfolio management, as the selected project portfolio determines the future development of the application landscape. Regarding the state of the art in the context of project portfolio management, most decisions about portfolios are currently based on *gut feel*, not on information, which is derived from a comparison of different *variants* of the landscape regarding quantitative or qualitative aspects (cf. [7]). The landscape variants therein should be related to the project portfolios, they result from. These variants have to be stored to facilitate comparisons and therefore be used to provide decision support.

EA management follows a typical management cycle consisting of the phases: Plan - Do - Check - Act (cf. [8,9]). Thereby, the traceability[2] of management decisions taken in the *Plan* phase and implemented in the *Do* phase, must be ensured to control the achievement of objectives (*Check*). An exemplary question in this context could be: Is the status of the planned landscape reached within the planned time frame or has the plan been changed? This information is subsequently used to prepare the next management cycle (*Act*). Consequently, a third type of information has to be stored in an information model for landscape management besides the *planned for* and the *variant* information: the moment in time the landscape was modeled (*modeled at*). From this discussion the following research question has been derived:

How should an information model for landscape management be designed to incorporate both business and technical aspects, and to support future planning and traceability of management decisions?

This question takes aspects of *temporality* as connected to landscape management into account. Therein, different versions of the landscape are of importance: the *current*, *planned*, and *target* version. The current landscape represents the status quo of the landscape *as is*. The planned landscape represents a future state of the landscape as *to be* at a specific time in the future[3]. This state is modeled by an architect at a certain time, emphasizing e.g. the changes performed by projects up to that specific future date. As a long term perspective the target landscape shows the architecture of the application landscape as envisioned at

[2] Traceability of decisions can be achieved by storing previous states of the managed objects. The respective technique is mostly referred to as *historization*.

[3] In some publications on EA management(cf. e.g. [13]), the terms *as-is* and *to-be* are used for the respective landscape version. We do not use this terminology, as the term *to-be* is often used ambiguously for both planned and target landscapes.

a certain time. Thereby, there is no need to have projects defined transforming the current or planned landscape into the target one. Furthermore, the target landscape does not necessarily specify deployed business applications but refers to envisioned future support providers.

Summarizing, the traceability aspects of landscape management lead to three different *time-related* dimensions: Firstly, a landscape is *planned for* a specific time, secondly, a landscape has been *modeled at* a certain time, and thirdly, different *variants* of a planned landscape may exist. Figure 1 illustrates the relationships between current, planned, and target landscape as well as the different dimensions relevant for landscape management.

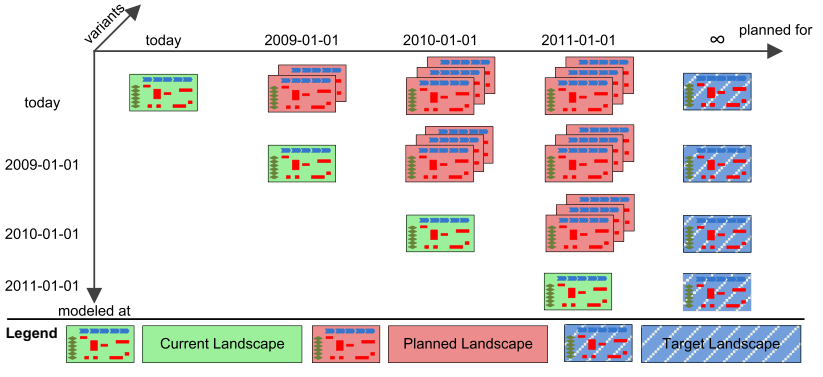


Fig. 1. Current, planned, and target landscape

The research question is approached in this article as follows: Section 2 gives an overview on current approaches to landscape management as described by researchers and practitioners in this field. Further, requirements – especially time-related ones – for an information model for landscape management are introduced. Thereby, a framework for the analysis of the support for landscape management is established. Alongside this framework an analysis of the current tool support for landscape management is performed in Section 3. Section 4 discusses ideas, which could be used to create an information model for landscape management fulfilling the aforementioned requirements. Therein, especially ideas originating from related modeling disciplines are considered. Finally, Section 5 hints at further areas of research in the context of EA management.

2 Requirements for and Current Approaches to Landscape Management

Due to the importance of managing the application landscape as a constituent of EA management, a number of different ways to approach this task have been

proposed both in practice and academia. Subsequently, we give an overview on these approaches with an emphasis on the aspect of temporality.

In [10] the application landscape is referred to as a concept specifying the enterprise's business applications and their interdependencies. This information is reflected in the information model of [10] via *interfaces* utilized to interconnect the *applications*. References from these application level concepts (on the *application layer* as in the notion of [10]) to business level entities, e.g. the different types of *business processes* (on the *organizational layer* of the model), are present and can describe the way, how business support is provided. The question at which organizational unit which business process is supported, by which business application, cannot be answered based on the information model. The aspect of temporality is also only partially addressed, while the models contain ways to store life cycle states of applications, it does neither support planning transitions between life cycle states nor does it take projects into account.

In [11] the business applications ands their relationships to other EA constituents form an important information asset, which should be presented to managers to provide decision support. As presentation form of choice, they introduce a type of visualizations, called *landscape maps*, in which the business applications are related to business functions and products. This relationship is referred to in [11] as a ternary one, which could also be established between applications and two other concepts, although such considerations are not detailed in the article. Temporal aspects are not part of the approach, while ways to use the landscape map visualizations for interacting and changing the data in the underlying models are explicitly stated.

A slightly different focus on managing the application landscape is taken in [12]. Therein, the aspect of the interfaces connecting the business applications is analyzed. The number of interfaces associated to a business application is considered an important impact factor, e.g. when changes to the application landscape are considered. In this context, [12] puts special emphasis on documenting and analyzing the current application landscape. This information is used as input to coordinate potential change processes affecting the landscape. While [12] takes a rather detailed look on the business applications and their interconnections, relationships to business related concepts of the EA are not presented in the paper. Whereas, the topic of the evolution of the application landscape is indicated, actual planning of future states or transformation projects is not discussed.

Beside the academic community also practitioners address the field of landscape management. In [5] the overall architecture of the application landscape is considered an important topic of EA management, exerting strong influence on the overall success of the company. Detailing the aspects of landscape management, [5] emphasizes on the relationships of the applications to the business processes, they support, as well as to logical structuring principles, e.g. organizational units. Further, the importance of application landscape planning is referred to, by complementing the current landscape with a target landscape,

not solely consisting of business applications but also of more abstract *providers of support* for business processes. Issues of how to transform from the current to the target landscape are also discussed in [5], although concept of planned landscapes is not introduced. Further topics, e.g. traceability of management decisions, are not considered therein.

In [13] the application landscape is presented as a management subject embedded in the context of business and technical concepts, ranging from business processes to technical platform modules. The current landscape should, accordingly, be documented with references to these aspects, especially the technical ones. Complementing the current landscape, a so called *ideal landscape*[4] should be defined as part of a landscape management endeavor, incorporating technical visions of the landscape. Mediating between current and ideal, different *to-be landscapes*[5] should be developed,of which each is assigned to a set of projects, which must be executed to realize the to-be landscape. Here, a strong relationship between the projects and the to-be landscapes is maintained, nevertheless means for tracing back the evolution of a to-be landscape are not incorporated.

Subsuming the state-of-the-art in managing application landscapes as presented in literature, different approaches are employed especially concerning the aspect of temporality. Nevertheless, creating an information model of the application landscape is a widely accepted prerequisite employed in landscape management. In some of the papers, presented above, information models are provided. These information models differ widely regarding the concepts introduced and the relationships as well as regarding their complexity. We regard, notwithstanding, such a model to be mandatory to approach landscape management as a whole and the important aspect of temporality in special.

Due to great interest from industry partners in information about EA management tools and especially their capabilities to address the concerns arising in the context of landscape management, an extensive survey – the *Enterprise Architecture Management Tool Survey 2008* – was conducted [14]. The survey was developed in cooperation with 30 industry partners. The survey pursued a threefold evaluation approach, relying on two distinct sets of scenarios together with an online questionnaire. Thereby, the first set of scenarios focuses on specific functionality, an EA management tool should provide, without connecting these functionalities to the execution of a typical EA management task, e.g. *1) flexibility of the information model* or *2) creating visualizations,*. The EA management tools are further evaluated by the scenarios of the second set, which reflect tasks that have been identified as essential constituents of many EA management endeavors, e.g. *1) business object management,* or *2) SOA transformation management*. One of the most prominent scenarios of the second part is the scenario *landscape management*, which is concerned with the managed evolution of the application landscape [15]. The concern of the scenario was described by the industry partners as follows:

[4] *Target* landscape in the terms used throughout this paper.
[5] In this paper, these landscape are called *planned* ones.

Information about the application landscape should be stored in a tool. Starting with the information about the current landscape potential development variants should be modeled. The information about the current application landscape and future states should be historicized to enable comparisons. [14]

Subsequently, a catalog of typical questions in the context of landscape management as raised by the industry partners is given:

- What does the current application landscape look like today? Which business applications support which business process at which organizational unit?
- How is, according to the current plan, the application landscape going to look like in January 2010? Which future support providers support which business process at which organizational unit?
- What was, according to the plan of 01-01-2008, the application landscape going to look like in January 2010?
- How does the target application landscape do look like?
- What are the differences between the planned landscape according to the plan of 01-01-2008 and the current plan?
- What projects have to be initiated in order to change from the planned landscape (according to the current plan) to the target landscape? What planning scenarios can be envisioned and how do they look like?

Based on the questions from the industry partners and the different dimensions relevant for landscape management, the following requirements regarding an information model can be derived, the model thus must:

(R1) contain a ternary relationship in order to support analyzes regarding current and future business support (which business processes are supported by which business applications at which organizational units),

(R2) provide the possibility to specify envisioned business support providers in order to facilitate target landscape planning without having to specify implementation details of the business support,

(R3) support the deduction of future landscapes from the project tasks, which execute the transition from the current to the future business support,

(R4) foster the creation of landscape variance based on distinct project portfolios in order to tightly integrate project portfolio management activities, and

(R5) ensure the traceability of management decisions by storing historic information of past planning states. This information may be interesting especially if complemented with information on the rationale for the decisions.

Based on these requirements, an overview about the support for landscape management as provided in the approaches from literature is given in Table 1.

3 Tool Support for Landscape Management

The solutions of nine major players in the market of EA management tools were analyzed regarding the information models, which they come shipped with. Three different exemplary approaches as taken by the different tools are subsequently explicated to provide an overview about the current operationalizations

of landscape management. The attributes are thereby not shown to improve readability but are mentioned in the description, if necessary for understanding.

Prior to discussing the different approaches taken by the tools, the core concepts of landscape management, which are most likely to be represented as classes in the information models, are briefly introduced (for details see [1]):

BusinessProcess: A business process can, according to Krcmar [16], be defined as a sequence of logical individual functions with connections between them. The process here should not be identified with single process steps or individual functions, but with high-level processes at a level similar to the one used in value chains. Thus, a process can have a predecessor process and a successor process, expressed by a respective relationship.

DeployedBusinessApplication: A deployed business application is a software system, which is part of an information system in an organization. The term refers to an actual deployment. In landscape management, business applications are restricted to applications that support at least one process.

FutureSupportProvider: A future support provider poses a sort of envisioned planning object, to be used instead of an actual deployed business application in a target landscape to define a business support.

OrganizationalUnit: An organizational unit represents a subdivision of the organization according to its internal structure, e. g. the entities showing up in an organigram can be used as organizational units.

Project: Adaptations of the application landscape are performed by projects, which each hold different attributes with temporal information, e.g. for their *startDate* and *endDate*[6]. Additionally, a project is *plannedAt* and *removedAt* referring to the planning time of its creation and of its deletion – effectively resulting in a time interval of validity, which is assigned to each project. A relationship between the project and the concepts affected by it, e.g. deployed business applications, exists.

SupportProvider: A support provider is an abstract concept, representing an entity, which can provide support for a business process at a specific organizational unit. In the context of the information model, actually deployed application systems can be used as *SupportProvider* instances as can future business applications.

SupportRelationship: This concept represents the support of a specific business process by a specific support provider at a specific organizational unit.

Starting with a basic approach to landscape management tool 1 presents an information model containing landscape management related concepts, as shown in Figure 2. Here, the business process is connected with the organizational unit via the support provider to support target landscape planning (cf. **R2**). Whereas data gathered according to this information model can support basic analyzes of the business support for a business process, the relationship to the organizational unit, where the support takes place, is not derivable unambiguously (cf. **R1**).

[6] If more detailed modeling of projects should be performed, the temporal information could be extended to contain starting and ending dates for different phases of the project, e.g. *planning* and *development*.

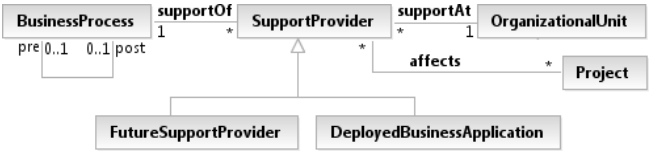


Fig. 2. Information model of tool 1

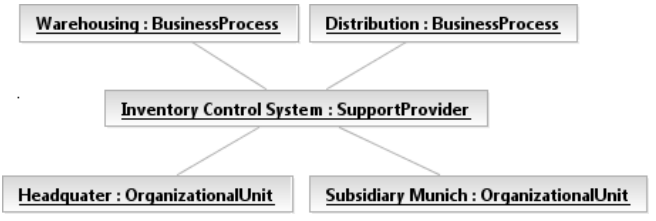


Fig. 3. Instance data corresponding to information model of tool 1

Figure 3 shows exemplary data instantiating the information model from Figure 2. Analyzing this data, a statement, which business process is supported by the *Inventory Control System* at the *Subsidiary Munich* cannot be made.

Besides the missing ternary relationship between business process, organizational unit, and support provider, the only concept carrying temporal information – the project – is connected to the support provider via the relationship *affects*. Thus, no time information for the business support provided can be stored (cf. **R3**). In addition, planning variants of the landscape can only be built based on the support providers instead of the business support provided (cf. **R4**). Consequently, tool 1 only rudimentarily supports the management of current, planned, and target landscapes. While such information might be sufficient for future planning in a one dimensional manner, the requirements concerning traceability and versioning cannot be addressed (cf. **R5**).

The information model of tool 2 (see Figure 4) incorporates the ternary relationship between the business processes, the organizational units, and the support providers by introducing a dedicated class and respective associations (cf. **R1**). The association *supportBy* is further assigned life cycle parameters using a mechanism similar to an association class in UML. Thus, it is possible to indicate that the business support provided by a specific instance of class *SupportProvider* is at a certain point in time in a specific life cycle phase, e.g. *planned* or *active* (cf. **R2**). This notion of life cycle is nevertheless disconnected from the concept of the project, which is independently associated to the class realizing the ternary relationship. While this association allows to model, that the support for a specific business process executed at a specific location is affected by a project, no mechanism to indicate, which *SupportProvider* actually is changed by the project, is present (cf.**R3** and **R4**)[7]. Further, the model does

[7] This fact is caused by the `*` multiplicity on the `SupportProvider` end of the `supportBy` association.

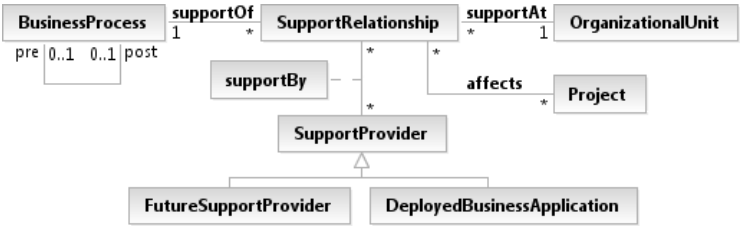


Fig. 4. Information model of tool 2

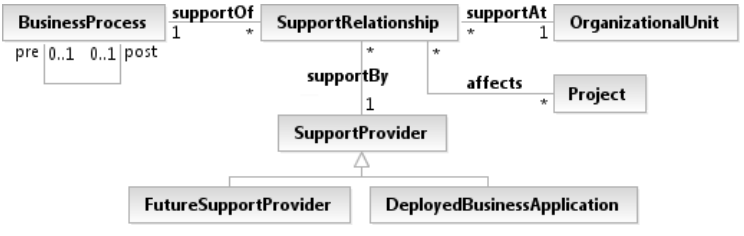


Fig. 5. Information model of tool 3

not support the creation of different landscape scenarios, as it is not possible to make projects or providers of business support belong together in one scenario. A mechanism for marking a *SupportProvider* an element of a target landscape is nevertheless provided via a flag attribute *target* in the association class *supportBy*. Historization of planned application landscapes is not supported (cf. **R5**) as no means for versioning instances corresponding to the model are given.

Finally, the information model of tool 3 is presented (cf. Figure 5), which is only slightly different from the model of tool 2, provides additional support for application landscape management – future state considerations are supported similarly as in tool 2 (cf. **R2**). The information model contains a support relationship, which supports analyses regarding the business support provided for a business process by a business application at an organizational unit (cf. **R1**). Nevertheless, the information model as proposed by tool 3 also implements temporality in a one dimensional manner by the project concept (cf. **R3** and **R4**), which affects the support relationship and contains temporal information, e.g. start and end dates. Such information might be sufficient for planning the evolution of the EA, but is somewhat limited concerning traceability of changes to the plans (cf. **R5**), which would demand support for *bitemporal* modeling. As an example, one might think of a plan for the EA regarding the year 2010, which might look different as-of begin 2008 respectively begin 2009.

Table 1 provides an overview about the evaluation results of the tool support for landscape management. Thereby, the support provided by the different approaches is indicated by symbols ranging from complete fulfillment of the requirement (●) via partial fulfillment (◑) to approaches, which totally lack support for the analyzed requirement (○).

Table 1. Existing Approaches and Tools and their Fulfillment of the Requirements

	[10]	[11]	[12]	[5]	[13]	Tool 1	Tool 2	Tool 3
R1	○	●	○	●	●	○	●	●
R2	◑	◑	◑	●	●	●	●	●
R3	○	○	○	◑	●	○	◑	●
R4	○	○	◑	○	◑	◑	○	●
R5	○	○	○	○	○	○	○	○

In order to discuss the constituents of an information model fulfilling the requirements, we subsequently detail on related modeling techniques.

4 Discussing an Information Model for Landscape Management

The question, how to incorporate aspects of time in a database system has been repeatedly discussed in scientific literature (see e.g. [17,18]). A simple approach is to introduce a time stamp attribute to the table, which should be enriched with temporal information. This allows to specify that an entry of the table is valid since the point in time specified by the time stamp. The approach has the disadvantage that it is not possible to specify that the information stored in the table row is valid for a certain period of time. In order to resolve this problem another attribute can be introduced to define up to which point in time the values are valid, thereby capturing the history of a changing reality.

If traceability should also be considered a so called *transaction time* has to be specified additionally to the *valid time*, which has been described before. According to [18] this can be done by introducing two additional attributes defining a time interval for capturing the sequence of states of a changing table. Such a table is than called a *bitemporal table*.

Similar discussions have taken place for object-oriented models. From these discussions, a few design patterns, of which [19] gives a good overview, have emerged. In addition, [19] introduces basic time concepts: event, time interval, and duration, of which the latter ones can be considered of special importance for our design issue, e.g. for modeling life cycle information of business applications.

Additionally useful in the context of creating temporality aware object-oriented models are the temporal (design) patterns presented in [20]. The concept of the temporal association introduced there can be utilized to model, that the objects referenced in association can change over time. The actual realization of the pattern introduces a class with a validity interval making the association explicit. Other design patterns for addressing temporality exist, e.g. the edition [20], but are not further discussed here.

In order to fulfill the requirements as mentioned in Section 2, especially **R4** and **R5**, which have not been well addressed by the majority of tools, temporal patterns, as alluded to above, could be utilized. This challenge can be met as a central relationship of the landscape management models – the ternary one relating support providers, business processes, and organizational units, which is

explicated as an independent class, the *SupportRelationship*. Thereby, the pattern of the *temporal association* [20] could be incorporated – the associated projects could help to supply periods of validity for the *SupportRelationship* instances, i.e. the referenced *SupportRelationship* becomes valid, once the *endDate* of the project is reached.

If landscape plans for the same point in time created at different times should be compared to each other (cf. **R5**), the information concerning the point, when the project has been planned at, had to be considered. Consistently, the temporal pattern *edition* (cf. [20]) could be used to implemented this mechanism.

5 Outlook

In this article, we discussed time-related issues of application landscape management and how they relate to other tasks in EA management, especially project portfolio management. Section 2 showed different approaches to landscape management as found in literature. Further, we discussed requirements for landscape management, as gathered from EA management practitioners during an extensive survey, and finally compared the findings from literature and practice. Section 3 discussed the tool support for landscape management with an emphasis on the underlying information models. From this, the drawbacks of the different approaches were explained. Related fields of modeling were taken into account in Section 4 discussing, how an information model could be created to fulfill the requirements. Therein, especially *temporal patterns* for object oriented models proved to be promising.

Two interesting directions of research result in continuation of the discussions undertaken in this paper. At first, the ideas for constructing a temporal information model for landscape management have yet not been validated. This would nevertheless be an important step. Thereby, the creation of such a model is likely to comprise additional difficulties, especially if considered in the context of a larger EA information model. Second, a temporal information model is due to its inherent complexity likely to be usable only via an appropriate tool. This is especially true considering the temporal relationships to be maintained by a user – constrastingly, currently no tool capable of managing such an information model in a convenient way exists.

The other research direction emphasizes that landscape management is not the only part of EA management that has strong related issues associated. Managing the infrastructure of the enterprise might be also influenced by different dimensions of time. Temporal aspects addressed in one part of EA management may also exert certain influences to other parts of managing the EA, which independently might not be affected by temporality aspects. In order to address this, the information model could be organized in patterns, which keep temporality related issues contained in one fragment of a larger model. A technique potentially helpful in this context is the EA management pattern approach [1].

References

1. Buckl, S., Ernst, A., Lankes, J., and Matthes, F.: Enterprise Architecture Management Pattern Catalog (Version 1.0, February 2008). Technical Report TB 0801, Chair for Informatics 19, Technische Universität München (2008)
2. Lankhorst, M., et al.: Enterprise Architecture at Work – Modelling, Communication, and Analysis. Springer, Heidelberg (2005)
3. Winter, R., Bucher, T., Fischer, R., Kurpjuweit, S.: Analysis and Application Scenarios of Enterprise Architecture – An Exploratory Study. Journal of Enterprise Architecture 33, 33–43 (2007)
4. The Open Group: The Open Group Architectural Framework Version 8.1.1 (cited 2008-10-03), http://www.opengroup.org/architecture/togaf8-doc/arch/
5. Dern, G.: Management von IT Architekturen (Edition CIO). Vieweg, Wiesbaden (2006)
6. Fischer, R., Aier, S., Winter, R.: Enterprise Modelling and Information Systems Architectures. In: Proceedings of the 2nd International Workshop EMISA 2007, St. Goar, Rhine (2007)
7. Lankes, J., Schweda, C.: Using Metrics to Evaluate Failure Propagation in Application Landscapes. In: Multikonferenz Wirtschaftsinformatik (MKWI) 2008, München (2008)
8. Deming, W.: Out of the Crisis. Massachusetts Institute of Technology, Cambridge (1982)
9. Shewart, W.: Statistical Method from the Viewpoint of Quality Control. Dover Publication, New York (1986)
10. Braun, C., Winter, R.: A Comprehensive Enterprise Architecture Metamodel and Its Implementation Using a Metamodeling Platform. In: Desel, J., Frank, U. (eds.) Gesellschaft für Informatik 2005, Bonn (2005)
11. van der Torre, L., Lankhorst, M., ter Doest, H., Campschroer, J., Arbab, F.: Landscape Maps for Enterprise Architectures. In: Dubois, E., Pohl, K. (eds.) CAiSE 2006. LNCS, vol. 4001, pp. 351–366. Springer, Heidelberg (2006)
12. Garg, A., Kazman, R., Chen, H.-M.: Interface descriptions for enterprise architecture. Science of Computer Programming 61, 4–15 (2006)
13. Engels, G., Hess, A., Humm, B., Juwig, O., Lohmann, M., Richter, J.-P., Voss, M., Willkomm, J.: Quasar Enterprise – Anwendungslandschaften serviceorientiert gestalten. dpunkt Verlag, Heidelberg (2008)
14. Matthes, F., Buckl, S., Leitel, J., Schweda, C.: Enterprise Architecture Management Tool Survey 2008. TU München, Chair for Informatics 19 (sebis), Munich (2008)
15. Aier, S., Schönherr, M.: Enterprise Application Integration – Flexibilisierung komplexer Unternehmensarchitekturen. Gito, Berlin (2007)
16. Krcmar, H.: Informations management, 4th edn. Springer, Berlin (2005)
17. Date, C.: An Introduction to Database Systems. Addison Wesley, Boston (2000)
18. Snodgrass, R.: Developing Time-Oriented Database Applications in SQL. Morgan Kaufmann Publishers, San Francisco (2000)
19. Anderson, F.: A Collection of History Patterns. In: Harrison, N., Foote, B., Rohnert, H. (eds.) Pattern Languages of Program Design. Addison Wesley, Boston (1999)
20. Carlson, A., Estepp, S., Fowler, M.: Temporal Patterns. In: Harrison, N., Foote, B., Rohnert, H. (eds.) Pattern Languages of Program Design. Addison Wesley, Boston (1999)

A Lightweight Method for the Modelling of Enterprise Architectures

Henk Koning, Rik Bos, and Sjaak Brinkkemper

Department of Information and Computing Sciences, University of Utrecht,
P.O.Box 80.089, 3508 TB Utrecht, The Netherlands
{h.koning,rik,s.brinkkemper}@cs.uu.nl

Abstract. This paper introduces an easy to learn method to describe enterprise architectures with a limited focus on the relation between enterprise functions and IT-systems. We have designed the Enterprise Architecture Modelling method (EAM), driven by our teaching requirements and based on our ERP modelling experience. EAM consists of these diagram types: the Supply Chain Diagram, showing the business environment; the Enterprise Function Diagrams for the interoperation of enterprise functions; the Scenario Overlay for modelling the main business processes; the System Infrastructure Diagram, depicting the technical infrastructure of IT systems and networks; and the Application Overlay Diagram, showing which applications give support to which enterprise functions. We satisfactorily conducted about 40 case studies based on EAM. To solicit feedback we performed an enquiry among users of EAM. A future step will be testing the use of EAM in managerial decision taking in practice.

Keywords: Enterprise architecture, enterprise functions, applications, modelling method, views, feedback.

1 Lightweight Enterprise Architecture

Several years ago we started at our computer science department of the Utrecht University with a master course in Enterprise Architecture (EA). Since this is a vast subject we were searching for an approach which at the one hand would give the students a good overview of all the developments in EA and on the other hand would enable the students to test their skills in this challenging field. We have sought a limited area within the broad field of EA where the students could develop their skills in communication, in the analysing, condensing and abstract modelling of lots of information, and in mastering abstract concepts in the business and in information technology. We have chosen to concentrate on the bridge between enterprise functions and the IT-support for these functions, and to stay at a high abstraction level. In this we are inspired by our experience in modelling ERP-applications. In that area we have seen that a high level functional breakdown of an organisation is a stable reference to chose ERP-modules. The questions we seek to answer in our EA modelling method for a given enterprise are: what are the main functions the enterprise performs? What are the relations of these functions to each other and to the

G. Feuerlicht and W. Lamersdorf (Eds.): ICSOC 2008, LNCS 5472, pp. 375–387, 2009.

outside world? What are, or should be in the future, the information systems that support the enterprise functions? What infrastructure, in terms of computers and network capabilities, is necessary, or will be necessary in the future, to operate these information systems? The emphasis is on understanding the business in a break down of business functions; from thereon the relations are uncovered to information systems (applications) and infrastructure (network and computers). Our method only touches the surface of application layer and infrastructure layer. Also the relations of the business functions to the outside world are described only in general. We limit ourselves to a top down analysis, creating *overview* and *insight*.

The word 'lightweight' in the title of this section should not be confused with 'easy' or 'not to be taken serious'. It can be very difficult to enter an unknown company and derive from all the information that is available a central, homogeneous, balanced model of the enterprise functions. Information systems can have many relations to each other, but not all relations are meaningful at the top, strategic business level. Eppler [3] has studied the communication problems between experts and managers, and has found, amongst others (page 15): "Summarizing these issues, we can conclude that experts struggle with three major issues when transferring their knowledge to managers: First, reducing or synthesizing their insights adequately, second, adapting these trimmed insights to the management context without distorting them, and third, presenting the compressed and adapted findings in a trust-building style and reacting adequately to management questions and feedback." EAM provides means in the synthesizing/compressing and adapting to management context, as will be explained in the sections 2 and 3, and is by no means easy. Also, the ability in EAM to decompose the Enterprise Function Diagrams (EFDs, see section 2) into several layers of detail, means that, if necessary, a lot of complexity can be captured in EAM.

1.1 Related Work

Several methods for developing enterprise architectures are available nowadays, e.g. ArchiMate [6], SEAM [14] and ARIS [10][11]. Each method has its own strengths and usually has its focus on one or more specific points, e.g. integration between different models, alignment between business and IT, business processes, communication etc. We mention some methods here briefly to compare them to our goals. At this moment we have chosen not to use any of these methods (mostly because of the time needed to learn the method) and to create our 'own space'. In the future, when our method has stabilized more, we may again look at this issue.

The ArchiMate project has produced an elaborate language for describing enterprise architectures. The conceptual model consists of 29 entities with a corresponding visual representation consisting of 40 symbols each having a specific meaning. It takes a while to get to learn each of the symbols and, in our experience, after some time of not working with the language one has to relearn the specific meaning of each of the symbols. Because we have only a few basic concepts and stay at a high abstraction level, we practically need no special icons in our method. The authors of ArchiMate have proposed 16 diagram types (viewpoints) as a basic set to work with the language, but many more could be constructed.

ARIS originated as a method for describing business processes in the context of SAP implementation processes, but has developed into a thorough, general purpose EA modelling tool. It is even more complex than ArchiMate. It has 5 basic views, but numerous diagram types to populate the views. On the summary page [10], p. 78, we count 23 'main' diagram types. The business process meta-model contains 300 entities and relations (p. 48). For *students* the time needed to master the basics of ARIS is not proportional to the analysis time needed to study the enterprise architecture of a company.

MEMO, Multi-perspective Enterprise Modelling, was developed by Frank [4]. He proposes a framework of three so called perspectives - strategy, organization and information system - each of which is structured by four aspects: structure, process, resources and goals. MEMO contains three modelling languages: strategy modelling language (MEMO-SML), organization modelling language (MEMO-OrgML) and object oriented modelling language (MEMO-OML), which allow for detailed modelling of the three perspectives. The EAM method we propose in this paper seems to be a subset of MEMO's strategy goals, organization structure and process, and information system resources. MEMO has an interesting setup, but it falls short of our wishes regarding 'easy to create' and 'easy to understand'.

Braun and Winter [1] describe an Enterprise Architecture meta model which has four layers: strategy, organization, application, software. The models of the first three layers are shown in a slightly simplified manner, and the relationships between these models are elaborated. A successful implementation of these models, using a meta modelling tool, is reported. The three simplified models contain respectively 27, 27 and 22 concepts. So, in our view, they are not easy to learn, they entice into a lot of detailing and underline the need we feel to start anew with a basic method containing few concepts. We like the limited number of layers, with the focus on linking the organization to the IT-support.

The well known framework of Zachman [12][14] has 36 different viewpoints to give aid in categorizing the architectural information, but gives no guidance regarding the modelling of the information. The same goes for an architecture process description like TOGAF (Open group 2002). It gives guidance regarding the activities of the architect, but does not have a modelling method. The 4+1 framework of Kruchten [5] gives an architectural structure and modelling method, but is more geared toward software architecture and not applicable to enterprise architecture.

The team of Wegmann of the École Polytechnique Fédérale de Lausanne has developed an object-oriented enterprise architecture method, called "Systemic Enterprise Architecture Methodology" (SEAM) [14]. As part of SEAM, they have developed the CAD tool "SeamCAD" [7]. SeamCAD enables the modelling of hierarchical systems (spanning from business down to IT) at different levels of detail (e.g. from large business transaction to detailed interactions). It has a philosophical underpinning which is not so easy to understand and doesn't give guidance as to what modelling concepts should be used or what levels should be distinguished. The notation is UML-like and some training is needed to read the diagrams.

1.2 Outline of the Paper

In the next section we present an overview of the EAM method with the key concepts. In section 3 we present each of the models of EAM and illustrate them with example

diagrams from a case study. In section 4 we outline our efforts to receive usage feedback by a large series of case studies and by a questionnaire. We finish the paper with conclusions and future work.

2 Enterprise Architecture Modelling Method

2.1 Diagramming Tools for Enterprise Architectures

The EAM method consists of the following diagrams:

A *Supply Chain Diagram* (SCD) shows how the enterprise works together with business partners to produce the goods or services for the customers (enterprise context).

An *Enterprise Function Diagram* (EFD) gives a top level breakdown of the main functions of an enterprise. The top diagram covering the complete enterprise is called the *corporate EFD*, and the lower level EFDs are called *function EFD*.

A *Scenario Overlay* (SO) shows how the enterprise functions in an EFD interoperate in a particular situation.

An *Application Overlay Diagram* (AO) shows which applications give support to which enterprise functions.

A *System Infrastructure Diagram* (SID) shows the main network topology, the main computers that function in the network and the main information systems that run on these computers to support the enterprise functions.

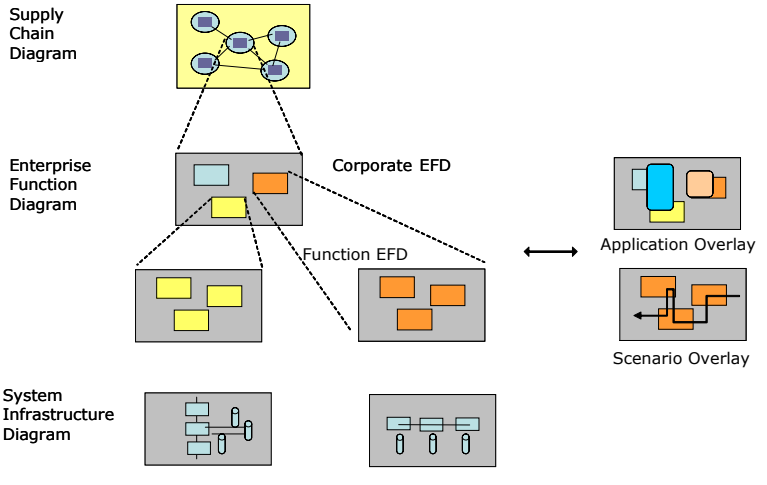


Fig. 1. Overview of EAM models

In Fig 1 the overview structure of these models is shown. We will explain these models each in turn in section 3. With each we give an example diagram taken from a case study performed at the company Center Parcs.

2.2 Concepts of Enterprise Architectures

See Fig 2 for the meta-model of our modelling method. Note that this somewhat simplified model does not contain all constraints as it only shows our key concepts and how they are related.[1]

For enterprise we use The Open Group [8] definition "Any collection of organizations that has a common set of goals and/or a single bottom line. In that sense, an enterprise can be a government agency, a whole corporation, a division of a corporation, a single department, or a chain of geographically distant organizations linked together by common ownership". For the sake of simplicity, when dealing with external parties, we include in this definition any collection of *individuals* (e.g. customers) that have a common set of goals and/or a single bottom line (and for which the enterprise to be modelled develops products and services).

To model the context of an enterprise we use enterprises connected by flows between their enterprise functions. For modelling the enterprise the key concept is the enterprise function. We define an enterprise function as a collection of coherent processes, continuously performed within an enterprise and supporting its mission. To show the interoperation of the enterprise functions we also portray the flow (of information or products & services). Scenarios indicate a sequence of flows. For modelling the information systems and the infrastructure our key concepts are computer, application and network (component). We use 'computer' as a general term to indicate all sorts of processing units or executional components. Likewise 'network' stands for all sorts of connectivity components. In section 3 we will describe the diagram types and indicate with each on what meta-model concepts they are based.

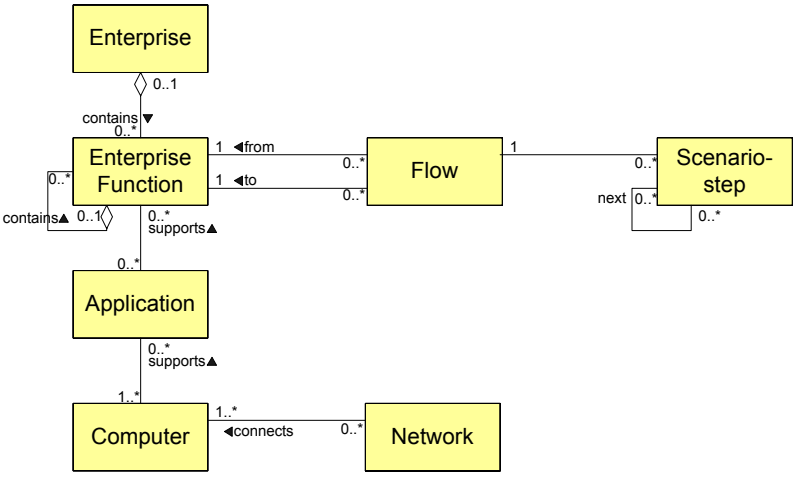


Fig. 2. The meta-model of the Enterprise Architecture Modelling method (EAM)

[1] Additional textual constraints are for example 'an enterprise function cannot send a flow to itself' and 'an enterprise function cannot contain itself'.

3 The EAM Models

In this section we present each of the models of EAM and illustrate them with example diagrams from a case study at Center Parcs Europe. Center Parcs Europe (CPE) is one of Europe's largest companies in the accommodations rental for short holidays. Its headquarters is in Rotterdam, the Netherlands and it offers about 10,000 bungalows and cottages in 20 parks.

3.1 Supply Chain Diagram (SCD)

The Supply Chain Diagram is a model of the *enterprise context* of the enterprise together with its business partners and the exchange of products and services. Supply Chain Diagrams create a quick overview of the enterprise as a whole and of the main players in its enterprise contexts. The SCD is based on the meta-model concepts 'enterprise', 'enterprise function' and 'flow'. In reality the flows connect enterprise functions in the different enterprises, but in the graphical presentation of the SCD these underlying enterprise functions are suppressed, see [2] for this technique. For large companies business units can be treated as separate enterprises. An EA description contains one SCD.

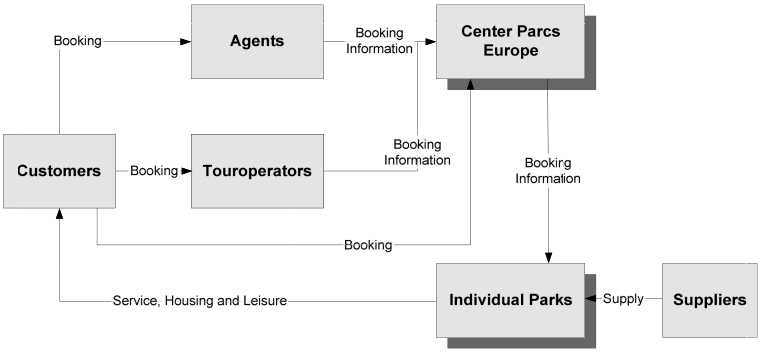


Fig. 3. SCD of Center Parcs

See Fig 3 for an example of an SCD. It shows how CPE and the Individual Parks cooperate with Agents, Tour operators and Suppliers to accommodate Customers. The boxes denote enterprises, the arrows the flow of products and services, or of information. In this example the two main units of CPE are shown, the central Europe headquarters and all the individual parks (in one box), and the main external parties.

3.2 Enterprise Function Diagram (EFD)

An Enterprise Function Diagram is a model from an *enterprise function* perspective. EFDs give a top level breakdown of the main operations of an enterprise with their information flows.

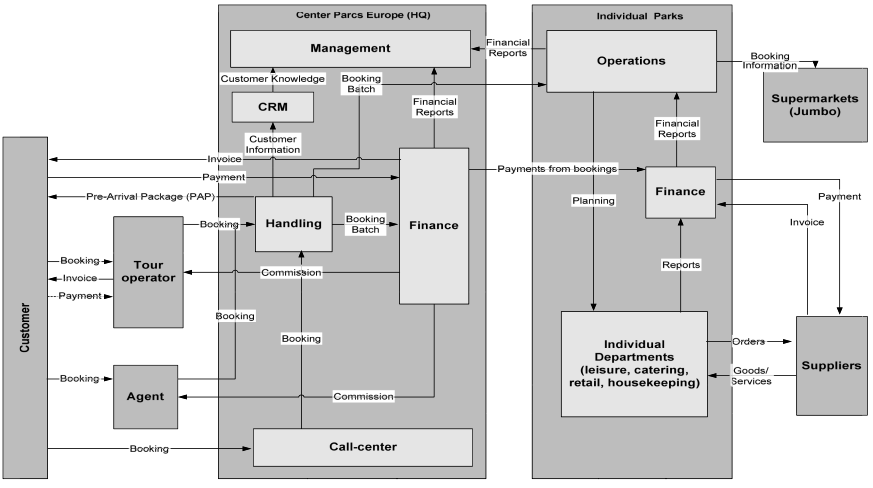


Fig. 4. EFD with the main enterprise functions of Center Parcs

See Fig 4 for an example of an EFD. The lighter grey boxes within 'Center Parcs Europe (HQ)' and 'Individual Parks' denote enterprise functions. To the left and to the right the external parties are portrayed. Arrows indicate the flow of information (digital or on paper). The EFD is based on the meta-model concepts 'enterprise', 'enterprise function' and 'flow'; this is the same as with the SCD but now the focus is on the enterprise functions. This diagram shows these functions of Center Parcs Europe (HQ): *Call center, Handling, Finance, Management, and CRM*. For the sake of brevity these are not described here further. These are the main functions of the Individual parks: *Individual Departments, Finance* and *Operations.*

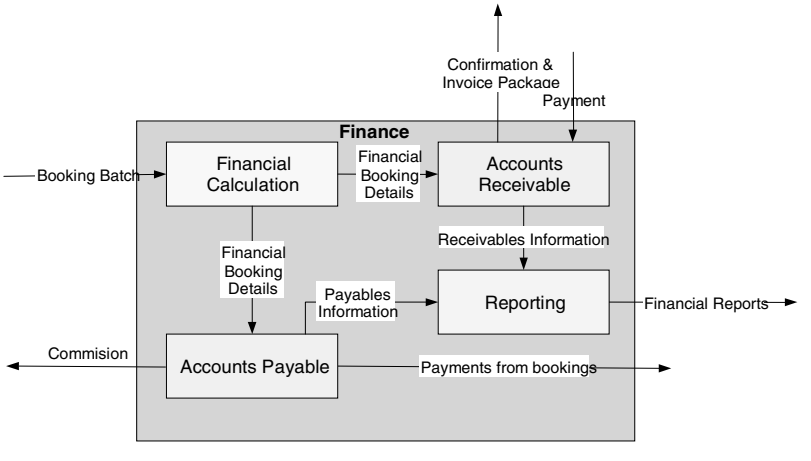


Fig. 5. Decomposition of the Finance function of Center Parcs Europe

Functional EFD. Enterprise functions in an EFD can be decomposed in the same diagram or in a separate EFD. See Fig 5 for an example of a decomposition. A tree structure of EFDs can be set up to analyze the architecture of an enterprise. Case study evidence (see section 4) shows that usually two levels are enough to get sufficient grip on the complexity of an organization. We call the top level EFD a *corporate* EFD (see Fig 4.) and a decomposition a *functional* EFD. The EFD of Finance shows the following sub functions. *Accounts Receivable , Accounts Payable, Financial Calculation,* and *Reporting*.

3.3 Scenario Overlay (SO)

A scenario is a continuous processing of a request trigger by various enterprise functions, which results in one or more feedback triggers. A Scenario Overlay provides insight in the interoperation of *enterprise functions* and in the completeness of the EFD. A scenario is drawn as an extra diagram level on top of an EFD with a proper explanation. Only essential flows are elaborated into a scenario (highest frequency, large impact). The scenario overlay adds limited, but for our goal sufficient, process information to the EFD. It gives fewer details than process models that have been created with a (dedicated) process modelling language. See Fig 6 for an example. Dark broad lines are drawn that touch EFD functions in the execution of the process that is triggered. The extra information for the scenario, compared to the underlying EFD, is based on the meta-model concept 'scenario step'. Here we show one scenario overlay for the Finance function. This scenario concerns a booking made via the CPE call center. Scenarios are an optional part of an EA description. Typically an EA description will contain several SOs.

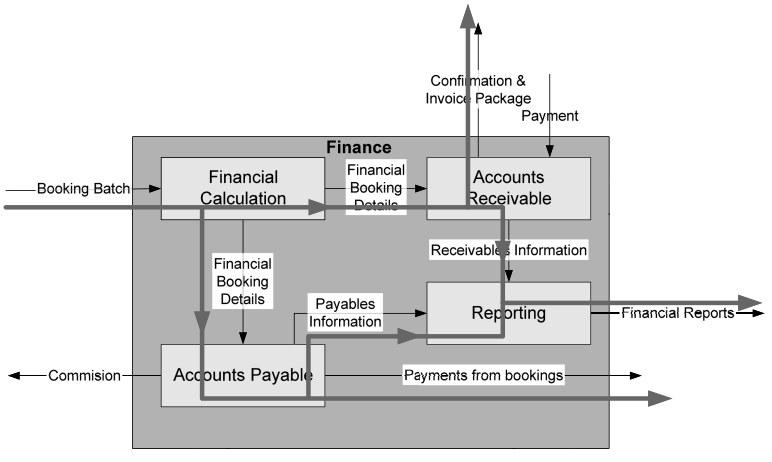


Fig. 6. Example Scenario Overlay, the financial processing of a Booking Batch

3.4 System Infrastructure Diagram (SID)

A SID shows the *information systems* and information technology *infrastructure* of an enterprise, or a well-defined part thereof. A SID shows the main network topology,

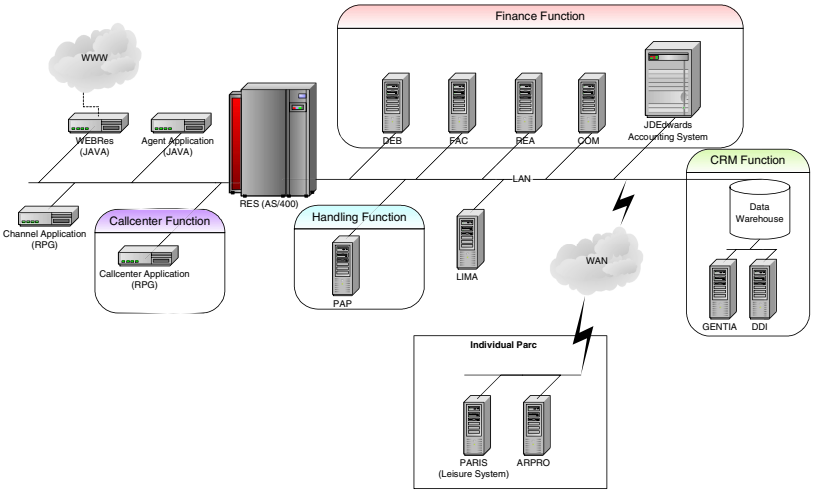


Fig. 7. Example System Infrastructure Diagram, a mixture of platforms at CPE

the main computers that function in the network and the main information systems (applications) that run on these computers to support the enterprise functions. The SID is based on the meta-model concepts 'application', 'computer' and 'network'.

The SID is not intended to manage the computers and network in their own right (that would need much more information), but only to show the technical context of the applications. For SIDs there are many popular notational variants. An EA description typically contains one SID, corresponding to the abstraction level of the corporate EFD. Complex parts in this SID can be exposed by creating a SID at a more detailed level. See Fig 7 for an example. The computer icons depict hardware systems, the adjoining acronyms stand for software systems (applications) that run on them. The hardware systems are grouped by the enterprise functions they support. The lines depict network connections. As can be seen in the infrastructure diagram, CPE uses many different systems.

3.5 Application Overlay (AO)

To fill a gap that is felt between the Enterprise Function Diagram and the System Infrastructure Diagram we later introduced a new type of diagram, comparable to the Scenario Overlay, the Application Overlay (AO). The applications are drawn as an extra diagram level on top of an EFD with a proper explanation. The AO deals with information systems (applications) in their own right. Information systems (software components) have a different lifecycle than computers and network (hardware components). The AO is based on the meta-model concepts 'enterprise function' and 'application'. See Fig 8 for an example.

This simple example was not part of the Centre Parcs case study, but was constructed by the authors of this paper as an example. The AO shows clearly which enterprise functions are supported, once or multiple times (possibly a sign of redundancy or fragmentation), and which functions are not supported by IT. If an

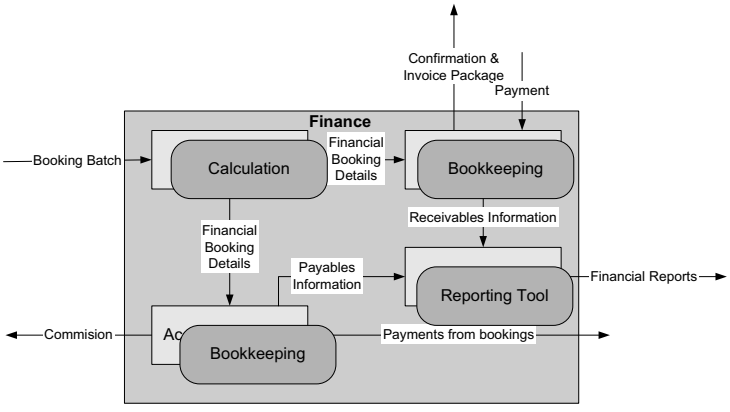


Fig. 8. Example Application Overlay , applications that support the Finance enterprise functions of Center Parcs Europe

application supports enterprise functions that are not adjacent in the underlying EFD diagram, then the application will be drawn more than once (see Bookkeeping in Fig 8).

4 Usage Feedback

4.1 Course Enterprise Architecture

Each fall a master course on Enterprise Architecture (see [13] for a general description) is given as a course in the international MSc program of Business Informatics at Utrecht University. An important assignment in this course is to do a case study in practice, meaning that the students, who already have a bachelor in computer science, have to produce a complete EAM model for a real life company, which includes all diagram types. A representative of the company has to consent to the models, and declare they really represented the given situation at the company. These representatives (contact persons) were mostly having a position in the IT-department, reporting to business management. In the first year we had 40 students in 19 groups working at 15 different companies. In the second year we had 68 students in 23 groups working at 18 different companies. Six companies participated for the second time, making up a total of 27 different organizations. The students are placed in groups of 2 or 3 each and before they visit the companies for the interviews, they get lectures on our EAM technique together with some small exercises. Lectures and exercises take about eight hours. Amongst these case studies we have seen many very good and insightful high level descriptions of enterprises and their IT-support.

EAM was used in practice by former students in several projects in the industry. Here we mention the following since these are published. First, at a municipality to decide on a new e-government portal service to be launched [16]. Second, the integration of enterprise applications at the Royal Netherlands Army was performed by applying EAM. EAM was *extended* with some UML diagrams to design the integrated business processes spanning multiple enterprise applications, see [9].

4.2 Questionnaire

We launched a questionnaire among the students who performed the case studies. See Table 1 for a summary of the results. Six questions related to the ease of creation were asked for each of the diagram types in EAM. The Application Overlay Diagram was not taken into account in this questionnaire, as it was introduced later. The students answered the questions immediately after performing the case study, and with respect to their own work. We have no comparable figures concerning other modelling methods and the students had no normative references for their answers. So we don't take the outcome of the survey as a vindication in an absolute sense, but we subjectively find the average scores satisfying (in line with similar questionnaires regarding other teaching subjects). We mainly look at relative differences in the figures or at extreme values to point us to aspects of EAM that need attention.

The most problems students experience seem to be with the EFD: the readability, the makeability and the correctness. We think this is understandable. The general tendency to add to much information to a diagram and to not stay at a high abstraction level is most felt with the EFD. Here also the intellectual effort needed to construct an abstract balanced model is felt most heavily. We see the same line in the figures pointing to "information lacking on all diagram types". In the presentation of EAM we need to stress more the limited scope e.g. the high abstraction level.

Table 1. Results of the questionnaire among students (n=23)

Question	Supply Chain Diagram	Enterprise Function Diagram	Scenario Overlay	System Infra-structure Diagram
Is the .. diagram easily readable? Rate on a scale from 1 (very bad readable) to 5 (very good readable) how good the .. diagram readable is.	4,3	3,2	3,7	3,8
Has the .. diagram the right level of abstraction? Make a choice: a. less detail preferred / b. just right the way it is / c. more detail preferred.	a 4% b 74% c 22%	a 9% b 91% c 0%	a 9% b 86% c 5%	a 14% b 54% c 32%
Is the correctness of the .. diagram easily established (conformity with the reality within the company)? Rate on a scale from 1 (very difficult) to 5 (very easy) how well the .. diagram can be checked.	3,5	2,7	3,5	3,4
Is there information lacking on the .. diagram? Chose y (yes, information is lacking) or n (no, no information is lacking).	y 35% n 65%	y 35% n 65%	y 23% n 77%	y 24% n 76%
Is there redundant information in the .. diagram? Chose y (yes, there is redundant information) or n (no, there is no redundant information)	y 4% n 96%	y 9% n 91%	y 5% n 95%	y 10% n 90%
How easy is it to produce this kind of diagram on the basis of available information? Rate on a scale from 1 (very difficult) to 5 (very easy) how well the .. diagram can be produced.	3,7	2,7	3,6	3,3

Contacts with some of the contact persons for the students at the participating companies, gave similar indications. Although they gave their consent that the produced EAM models truly reflected the actual situation at their company, they liked to have more information in the diagrams. We did not come round to asking more precisely what more information was needed in their view. The readability of the diagrams was considered good. No training was needed to read the diagrams.

5 Conclusions and Future Work

In this paper we have presented the Enterprise Architecture Modelling (EAM) method consisting of five diagram types for modelling enterprise architectures at a high level of abstraction in a fast and simple way. 40 Different case studies were performed using EAM. We conducted a small scale questionnaire. We conclude that EAM is a good means to express the essential functioning of an enterprise and it's IT-support. For authors EAM can be learned and trained in a short course of one day, but they find it sometimes difficult to stay at the high abstraction level in the application of the method. The resulting diagrams can be understood without any specific training.

Further research in practice is needed to assess the necessity and sufficiency of the diagram types for, for instance, managerial decision making. It is required to incorporate non-IT staff in future evaluations of EAM We want to continue developing EAM and take into account the attention points coming out of questionnaires. For the authors the creation of the EFD needs attention.

Besides these points we want to produce tool support for EAM (an alternative here is possibly to use an existing tool in a customized manner), and, together with a partner from industry, we want to develop the practical application of (an extended?) EAM.

What started as a mere conviction has now been tried out on a modest scale and usage feedback has been received. We would like to see EAM used on a larger scale in the teaching of EA and in practice, which will give us hopefully more feedback on the strengths and weaknesses of EAM.

Acknowledgements

We thank Ronald Bos and Inge van de Weerd for conducting the case study at CPE.

References

1. Braun, C., Winter, R.: A Comprehensive Enterprise Architecture Metamodel and Its Implementation Using a Metamodelling Platform. In: Desel, J., Frank, U. (eds.) Enterprise Modelling and Information Systems Architectures, Proc. of the Workshop in Klagenfurt, GI-Edition, Klagenfurt, 24.10.2005, Gesellschaft für Informatik, Bonn, P-75. Lecture Notes (LNI), pp. 64–79 (2005)
2. Buuren, R., van, H., Jonkers, M.-E., Strating, P.: Composition of relations in enterprise architecture models. In: Ehrig, H., Engels, G., Parisi-Presicce, F., Rozenberg, G. (eds.) ICGT 2004. LNCS, vol. 3256, pp. 39–53. Springer, Heidelberg (2004)

3. Eppler, M.J.: Knowledge Communication Problems between Experts and Managers. An Analysis of Knowledge Transfer in Decision Processes, ICA Working Paper #1/2004, University of Lugano, Lugano (2004)
4. Frank, U.: Multi-perspective Enterprise Modelling (MEMO) - Conceptual Framework and Modelling Languages. In: Proceedings of the annual Hawaii international conference on system sciences (2002)
5. Kruchten, P.: Architectural blueprints – The '4+1' View Model of Software Architecture. IEEE Software 12(6), 42–50 (1995)
6. Lankhorst, M., et al.: Enterprise Architecture at Work. Springer, Berlin (2005)
7. Le, L.S., Wegmann, A.: SeamCAD 1.x: User's Guide (2008) (accessed July 5, 2008), http://infoscience.epfl.ch/getfile.py?mode=best&recid=55774
8. The Open Group. TOGAF "Enterprise Edition" Version 8.1 (2002) (accessed July 5, 2008), http://www.opengroup.org/architecture/togaf8-doc/arch/
9. Roques, J., Vader, H., Bos, R., Brinkkemper, S.: SAIM - A situational method for application integration. UU-CS (Int. Rep. 2007-022). UU WINFI Informatica en Informatiekunde (2007)
10. Scheer, A.W.: ARIS – Business Process Frameworks. Springer, Berlin (1998a)
11. Scheer, A.W.: ARIS – Business Process Modelling. Springer, Berlin (1998b)
12. Sowa, J.F., Zachman, J.A.: Extending and formalizing the framework for information systems architecture. IBM Systems Journal, vol 31(3) (1992)
13. Utrecht University. Course Enterprise Architecture (2008) (accessed July 5, 2008), http://www.cs.uu.nl/education/vak.php?vak=INFOEIA&jaar=2007
14. Wegmann, A.: On The Systemic Enterprise Architecture Methodology(SEAM). In: Proceedings ICEIS 2003 (2003)
15. Zachman, J.A.: A framework for information systems architecture. IBM Systems Journal 26(3), 276–292 (1987)
16. Zuiderhoek, B., Otter, A., Bos, R., Brinkkemper, S.: Framework for Dutch Municipalities to Ensure Business IT Alignment Using Enterprise Architecture. In: Proceedings of the 6th European Conference on e- Government, pp. 457–466. Academic Conferences International, Reading (2006)

A Contingency Approach to Enterprise Architecture Method Engineering

Christian Riege and Stephan Aier

Institute of Information Management, University of St. Gallen,
Müller-Friedberg-Strasse 8
9000 St. Gallen, Switzerland
{christian.riege,stephan.aier}@unisg.ch

Abstract. Enterprise Architecture (EA) and methods for the design and employment of EA significantly contribute to the transparency, consistency and eventually to the flexibility of an organization. However, there is hardly any "one-size-fits-all" EA method that is equally effective for a broad range of transformation projects or in a large number of different contexts. Based on an empirical analysis, this paper identifies three relevant EA contingency factors as well as three dominating EA application scenarios as a basis for a situational EA method engineering taking these into account.

Keywords: Enterprise Architecture, Method Engineering, Contingency Factors.

1 Introduction

Enterprise architecture (EA) describes the fundamental structure of an enterprise [28, 33] and supports transformation by offering a holistic perspective on as-is as well as to-be structures and processes [19].

EA is widely accepted as an approach to manage transformations and to foster IT/business alignment [7, 22, 30]. In order to guide this transformation, EA methods are needed. While there are a number of EA methods available, e.g. [21, 31], a classification of methods is needed in order to understand in which situation a certain method is appropriate, how a method should be adapted to a certain situation, or for which situations new methods have to be developed. This is based on the assumption that there is no "one-size-fits-all" method, but that, depending on a certain situation, different methods—or at least different configurations or adaptations of a method—are needed. In order to develop an understanding of such situations, relevant contingency factors need to be identified which have an impact on the realization of EA.

As a foundation for situational EA method engineering and to continue the discussion started in [1] this paper further particularizes current realization approaches of EA. EA method engineering is an evolving discipline, which additionally requires outlining typical EA application scenarios. For this purpose our contribution is based on an exploratory analysis shaping current EA approaches, EA contingency factors as well as important EA application scenarios.

The remainder of this paper is organized as follows: Section two provides an overview of the theoretical background and related work. The discussion of the

G. Feuerlicht and W. Lamersdorf (Eds.): ICSOC 2008, LNCS 5472, pp. 388–399, 2009.

contingency factors in situational method engineering is reflected, and a short review on the state-of-the-art of enterprise architecture is given. Section three describes the details of the explorative empirical analysis aiming at identifying current EA realization approaches and section four outlines typical EA application scenarios. The paper ends with a conclusion and an outlook on future research activities in section five.

2 Theoretical Background and Related Work

Enterprise architecture literature provides a broad range of results that may be grouped into three categories. Category one comprises enterprise architecture frameworks. Popular examples are the Zachman Framework [35] and The Open Group Enterprise Architecture Framework—TOGAF [28]. Category two is comprised of publications by scientists as for example [8, 14, 19]. The third category is defined by practitioner's publications who predominantly publish for practitioners. Examples are [24, 29]. However, the boundary between scientific and practitioner approaches (regarding authorship as well as readership) is often fluid. Examples are [3, 22, 27].

A fundamental method provided by almost all of the contributions cited above is comprised of a basic approach for documenting the as-is and/or to-be EA. Some approaches provide a number of analyses that may be employed in an EA transformation method [19, 20] or a list of EA application scenarios for which methods may be developed. However, this list is neither complete nor are its items disjunctive.

Discussion in the field of EA is highly concerned with questions as which artefacts belong to EA, e.g. [2, 4, 15]. Only recently, it is discussed how to maintain EA models [7], how to use EA, or what benefits EA may provide to an organization [25]. Especially the latter issues require sound methods. Although there are isolated EA methods taking the situation of application into account, e.g. [34], there is no overall landscape of EA methods available.

A method may be defined as a systematic aid that guides the transformation of a system from an initial state to a target state. It is unlikely that there is an EA method, which fits to every problem situation in the field. Instead it is advisable to adapt an existing method or to use dedicated parts, like method components or method fragments. Approaches like this are discussed as situational method engineering [12, 18, 26]. It means that a method can be customized for the needs of a project or an organisation. In order to customize a method for a situation contingency or situational factors are needed to facilitate the description of such a situation. Existing contingency approaches are not tailored for EA and their contingency factors often lack empirical evidence [13, 17, 26]. Therefore the aim of this paper is to identify contingency factors determining current EA realization approaches by means of empirical analysis.

3 Current Realization Approaches of EA

An exploratory analysis was conducted in order to identify different EA approaches in practice [1]. The data was collected by means of a questionnaire filled in by participants of two practitioner conferences in 2007. Both conferences focused on EA in particular. Attending were IT executives, IT service providers and consultants as well as EA experts. In advance of the conferences, the questionnaire was subject to a pretest carried out by a group of enterprise architects.

3.1 Characteristics of the Data Set

A total of 69 questionnaires were returned. If the data set was incomplete regarding one of the 15 items used in subsequent analysis, the questionnaire was discarded. After applying this selection criterion, 55 valid questionnaires were analyzed. Although the sample size is rather small, the data set is considered adequate to provide a basis for an exploratory analysis.[1] The observed organizations mainly represent mid-size and large companies from the financial services sector as well as software vendors and IT consultants. In addition to demographic characteristics the data set comprises variables which can be divided into four groups and characterized as follows:

Constitution of EA: Architecture in general includes a set of IT artefacts like hardware assets, software components, and applications and extends the focus to business related artefacts. To ensure that business/IT alignment is adequately supported, EA also spans artefacts like business processes, products, customer segments, etc. Due to the large number of potential artefacts, EA is requested to represent the essential parts of the organization [35]. The data set contains information regarding the aforementioned variables.

Application scenarios and analysis techniques of EA: The employment of EA in an organization often refers to a substantial number of possible applications [19, 20, 32]. Applications however are external to the EA approach. The aim is to integrate EA into the organization's initiatives to secure that the organization develops in accordance with the structures defined in EA. For this reason, the EA model is subject to a range of analysis techniques. Techniques reveal dependencies between different EA artefacts, identify gaps or redundancies (e.g. application support of certain business processes), and reveal artefacts that might interfere with a homogeneous EA structure [19, 20, 32].

Maintenance of EA: This part of the data set contains information to which extent EA models are part of strategic planning, and to which extent EA models support transformations. Furthermore it covers the approach how EA data is gathered and maintained within an organization. A central instance for EA-related information facilitates a less complex and consistent EA improvement. In this holistic approach, a "leading" EA model is maintained covering all artefacts used to describe the EA. A federated approach puts more emphasis on specialized architectures and their models. The EA model is then supplied with data through periodically performed replications. EA data which is maintained via local repositories yields more flexibility, but also ensures that the stored information is up-to-date [7].

Communication and organizational structure of EA: On the one hand, the data set contains information on organizational roles which should be established to ensure EA is adequately represented within the organization—e.g. the role of an expert in EA modelling. On the other hand, EA offers benefits that take effect across IT and business units. It is important to capture how the concept of EA spreads within the organization. According to the understanding that EA is also involved in management

[1] What rule of thumb to use for factor analysis in determining an allowable ratio of cases to variables is still a matter of research taste [23]. However, [5] suggests a 4-to-1 rule of thumb for an allowable ratio of cases to variables.

activities and addresses business related objectives, it is of high importance how EA is perceived [14]. The information in this part of the questionnaire also covers the integration of EA processes into the organization's governance structure. The respondents were asked to assess the current degree of realization of each item in their organization. Therefore, the questionnaire chooses a five-tiered Likert scale. The minimum value (1) that was possible to check represents "nonexistent", whereas the maximum value (5) indicates an "optimized" realization.

3.2 Identifying Contingency Factors of EA

In order to identify contingency factors of EA, a factor analysis is applied. A factor analysis involves extracting a small number of latent factors among the variables in the data set. To form an adequate foundation, the data set has to meet two criteria. The first criterion is derived from the variables' anti image covariance. The anti image covers the part of the variance which cannot be explained by the remaining variables in the data set. As factor analysis aims at finding latent factors based on the data set, a data set is suitable for factor analysis if the anti image is rather low. According to [6], the percentage of none diagonal elements of the anti image covariance matrix, which are non-zero (>0.09), should not exceed 25%. In the case presented here, this parameter is about 17%. The second criterion involves the computation of the Kaiser-Meyer-Olkin measure of sampling adequacy. In the data set at hand, the measure is 0.798. According to [16], it puts a data set with a value of 0.7 or above into "middling" range, bordering the "meritorious" range. In this case, the results proof that the data set is generally appropriate for factor analysis. The factor analysis was performed based on a reduced data set of 15 items. While some items which are excluded from subsequent analyses relate to company properties such as staff size and industry sector, others were previously characterized as covering the constitution of EA within an organization.

As extraction method the principal component analysis was applied. Principal component analysis identifies few independent factors that contain the fundamental aspects of the data. In order to identify the optimum number of factors, the eigenvalue was computed which represents the amount of variance accounted for by a factor. According to the Kaiser criterion a factor's eigenvalue should exceed a value of one [10]. As a result, three contingency factors that account for 64% of the total variance were extracted. In order to better interpret the nature of the factors, the component matrix was rotated applying the Varimax method with Kaiser normalization. Each of the three factors consists of five items and can be described as follows.

Table 1. Factor 1—Adoption of advanced architectural design paradigms and modelling capabilities

Item 1.1 EA is developed with regard to modularization as an architectural design paradigm.
Item 1.2 The principles of service orientation form a basis on which EA is designed.
Item 1.3 EA models represent the current structure of the organization.
Item 1.4 Documentation of EA models includes target architecture.
Item 1.5 EA models support transforming EA from as-is structure towards to-be structures.

The items that load on contingency factor 1 describe valuable ways to adopt the concept of EA. On the one hand, it involves well established architecture design paradigms which emphasize the layered structure of EA. The findings denote that developing EA needs a certain degree of decoupling between the different EA layers as indicated by the principles of service orientation and thus foster re-use of EA artefacts. On the other hand, factor 1 points out that a further enhancement of EA also depends on the dimension of the EA documentation. To allow for a continuous development, not only loosely coupled artefacts, but also an idea of how to approach a future development stage is decisive. EA then contributes to business/IT alignment by offering simulation capabilities, which presupposes different variants of its to-be structures.

Table 2. Factor 2—Deployment and monitoring of EA data and services

Item 2.1	EA is measured and/or reviewed on a regular basis.
Item 2.2	Processes concerning EA management are subject to regular reviews.
Item 2.3	The role of an EA quality manager is established fostering and communicating EA concerns.
Item 2.4	EA is aiming to improve the overall homogeneity of architecture elements by applying heterogeneity analysis.
Item 2.5	EA is used to perform coverage analysis in order to illustrate redundancies or gaps regarding EA artefacts.

Factor 2 describes the deployment of EA within the organization. It is required to establish a consistent monitoring of EA data and services to further enforce the deployment. This can be assisted by the role of an EA quality manager who is responsible for observing periodic reviews of EA data and EA processes. A high degree of EA deployment puts the organization in the position to reduce its costs for maintenance activities, software and hardware licenses, but also to ensure that similar concerns are treated equally and according to the parameters of the EA roadmap. A high factor value also points to the application of sophisticated EA analysis techniques within the organization.

Table 3. Factor 3—Organizational penetration of EA

Item 3.1	EA is perceived as being valuable to the business units.
Item 3.2	IT departments explicitly refer to EA as a helpful instrument.
Item 3.3	IT departments use EA data in broad range of use cases.
Item 3.4	Business units base their work on EA data.
Item 3.5	EA data is part of the decision support for management units.

The third contingency factor accounts for the penetration of EA in the organization. The findings suggest that the overall level of penetration is influenced by the degree EA results and EA documentation are used by a broad range of stakeholders. According to this analysis, EA is a suitable tool not only to support IT related work, but also to serve the business units and to provide reliable information to management units. The findings suggest that as the level of organizational penetration increases with the organization's capability to clearly communicate EA benefits to the potential stakeholders—regardless if they actually operate on EA results or not. Therefore, the third factor describes the way EA is perceived and utilized across the organization. A high

level of organizational penetration leads to a higher acceptance, and less misinterpretation of EA within the organization, respectively.

3.3 Clustering EA Realization Approaches

In order to point out how EA is actually realized, the data set was partitioned into different subsets by means of a hierarchical cluster analysis. As input data, the factor values of the three aforementioned contingency factors were used. Ward's method has been used as clustering algorithm. It combines the two clusters which lead to a minimal increase in the within-cluster sum of squares with respect to all variables across all clusters. The squared Euclidean distance was selected as distance measure to determine the similarity of two clusters. Although the application of alternative measures may lead to different clustering results, the squared Euclidean distance was chosen as it is the most commonly recognized procedure [10] and moreover provides a comprehensible representation with respect to the sample's data structure. To gain information about the cohesiveness of clusters, a tree diagram—designated as dendrogram—serves as visualization and helps to assess the appropriate number of clusters to keep. There is no standard selection procedure to derive the number of clusters [10]. As the applied fusion algorithm aims at minimizing the within-cluster sum of squares in each step, it is appropriate to keep the number of clusters if the subsequent clustering step accounts for the highest increase of the total sum of squares [9]. In the analysis at hand, this heuristic suggests to distinguish between three clusters which in turn represent three different EA approaches. Table 4 exhibits arithmetic means ($\bar{x}$) and sample standard deviations (s) of the calculated factor values for each of the three clusters. A high value implies a high degree of realization among the cluster members regarding the factor items that load on the respective factor.

Table 4. Arithmetic mean and standard deviation of factor values

	Contingency Factor 1		Contingency Factor 2		Contingency Factor 3	
	$\bar{x}$	s	$\bar{x}$	s	$\bar{x}$	s
Cluster 1 (n =15)	1.24	0.74	0.26	1.11	0.29	0.95
Cluster 2 (n =10)	-0.20	0.83	0.62	1.26	-1.33	0.53
Cluster 3 (n =30)	-0.55	0.51	-0.34	0.70	0.30	0.77

Based on the information depicted in Table 4, the three clusters can be visualized by positioning them in a three dimensional coordinate system (Fig. 1). The horizontal axis of the coordinate system is represented by the factor *adoption of advanced architectural design paradigms and modelling capabilities*, the vertical axis displays factor 3 *organizational penetration of EA*. The Factor *deployment and monitoring of EA data and services* spans the third dimension. The clusters are arranged according to their arithmetic mean (cf. Table 4). To estimate the mean of the population when the sample size is small it is suggested to calculate the confidence interval that is derived from the Student's t-distribution [11]. For this purpose the confidence interval was calculated for each cluster based on the respective mean factor values (cf. Fig. 1). As a result the three cuboids visualize that each cluster differs significantly from another cluster in at least one dimension.

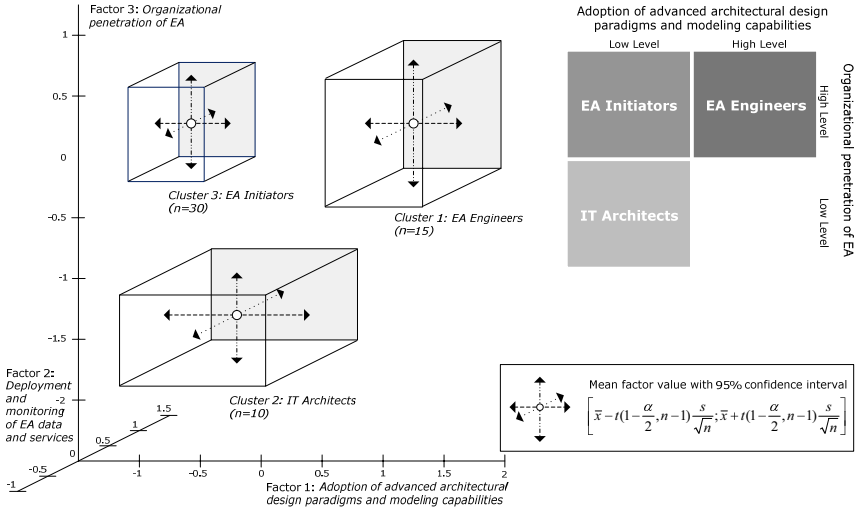


Fig. 1. Enterprise architecture realization approaches

Fig. 1 also exhibits the corresponding two-dimensional classification matrix, which excludes factor 2 as it does not account for significant cluster distinction. The matrix illustrates the distinct EA realization approaches considering factors 1 and 3. For both dimensions, high and low level are distinguished, which refer to either high or low parameter values. The realization approaches of EA can be characterized as follows:

Cluster 1: All 15 organizations which are assigned to this cluster, are characterized by sophisticated implementation of architectural design paradigms. They understand EA as instrument to represent a current structure of the organization, but also to deliver a roadmap for a future structure. It is reasonable to assume that organizational penetration is rather advanced among the members of the cluster. They are using EA rather as IT instrument, but also as a means of communication with the business. The organizations which belong to this cluster constitute an EA approach which may be designated as “EA Engineers”. EA engineers understand EA as a valuable instrument to develop and thus transform EA in its holistic understanding. They can also rely on a progressive perception of EA within the business and management units. EA engineers in its current state have an intermediate maturity regarding the employment and monitoring of EA data and services (factor 2).

Cluster 2: The second cluster is made up of 10 organizations which have a low level of both the organizational penetration of EA and the adoption of advanced architectural design paradigms and modelling capabilities. This combination can be characterized as observant attitude regarding a holistic EA. In this case, EA focuses primarily on IT architecture and, therefore, EA data is basically used in traditional IT project development. The relatively high value regarding the second factor supports this characteristic as it indicates a high deployment of (IT related) EA data. The EA approach represented by the organizations which are merged in the second cluster can be designated as “IT Architects”. They are well anchored in the IT domain. However,

this limited architectural understanding is an obstacle in order to really leverage the value of available IT understanding, models and methods. Rather advanced architectural design paradigms—e.g. service orientation—are not much developed in this cluster because they require a certain amount of organizational penetration.

Cluster 3: A total of 30 organizations are grouped into the third cluster. They are characterized by a high level of organizational penetration of EA—comparable with cluster 1. It is therefore reasonable to assume that the potential benefits of EA are recognized among these organizations. EA is understood not only as IT architecture, but also as an instrument to foster the alignment between IT and business. However, EA primarily focuses on documentation. Organizations which belong to this cluster can be designated as "EA Initiators". EA initiators put emphasis on transparency as the necessary precondition to realize benefits from EA application. Therefore, it seems reasonable to conclude that EA initiators in particular are interested in implementing relevant applications to demonstrate these benefits. This also explains the need for more sophisticated analysis techniques—which EA initiators lack of. This typically is a hint for a tool driven or model driven EA approach as opposed to an application driven approach. Such a tool driven approach may be dangerous since it requires significant efforts to survey and model the architectural data without a clear idea of future application scenarios.

The size of the clusters (Table 4) leads to the assumption that most organizations acknowledge the benefits of EA as EA initiators account for more than 50% of the three EA scenarios. Still a minority of organizations represented by the cluster IT architects is not able to convince potential stakeholders of EA benefits and thus is not able to leverage advanced design or modelling capabilities. The EA scenario with the currently most mature application of EA is represented by EA engineers.

4 EA Applications Scenarios

To adequately support EA method engineering, it is not sufficient to take contingency factors (c.f. section 3) into account but also to describe future EA application scenarios. In order to identify these scenarios, a second factor analysis has been performed based on 12 EA applications. The set of EA applications is derived from [20, 32]. Factor analysis serves as a means to reduce the dimensionality of that number of EA applications to a fewer number of factors. In contrast to the analysis in section 3.1, the respondents were asked to asses the future importance of each of the 12 suggested EA applications. As quality measures, the anti image covariance matrix as well as the Kaiser-Meyer-Olkin criteria were computed. In the data set at hand the percentage of none diagonal elements of the anti image covariance matrix is about 22%, well in range with the limit of 25% [6]. The Kaiser-Meyer-Olkin measure of sampling adequacy is about 0.828, putting the data set in the "meritorious" range [16]. The results assure that the data set is appropriate for subsequent factor analysis. Principal component analysis extracts three independent factors, which inherit the aspects of the 12 underlying EA applications. In total the three factors, representing three scenarios for EA application, account for 67.6% of the variance. Factor 1 consists of 4 items (Table 5) and can be characterized as follows.

Table 5. Factor 1—Support of Business Strategy Development

Item 1.1	Corporate Strategy Planning
Item 1.2	Business Process Optimization
Item 1.3	Quality Management and Improvement
Item 1.4	Business Continuity Planning

Factor 1 describes EA applications that affect the strategic development of an organization. EA supports decisions which e. g. demand reengineering business functions due to a potential shift in market requirements like quality aspects or processing time. Strategy development involves further analysis like a feasibility analysis to offer certain product bundles. EA is used to identify market offers/products that will be affected in case a specific application fails. In terms of business process optimization EA is a means to discover redundant processes which contribute to the same market offers/products. As an instrument being supportive for business strategy development, EA will need to revert to certain artefacts within the organisation. It is therefore necessary to have transparency e. g. regarding market segments, product catalogues, and business functions of the organisation and their interdependencies.

Table 6. Factor 2—Support of IT Management

Item 2.1	IT Consolidation
Item 2.2	Application Portfolio Management
Item 2.3	IT/Business Alignment
Item 2.4	Adoption of COTS
Item 2.5	Architecture Compliance Assessment

Factor 2 comprises of 5 items, (Table 6). Items which load on factor 2 specifically support IT Management within an organisation. Within this scenario EA is concerned with e. g. the advancement of the application landscape. EA serves as a means for analyzing the lifecycle of applications, and for example evaluate alternative replacement decisions. This is particularly important, if COTS is going to replace existing parts of the application landscape. Furthermore IT Management uses EA as a tool to consolidate the application landscape e. g. by analyzing whether there are applications without assigned users. Regarding IT Project Management, EA documents and provides an overview of compliance concerning technical or platform standards. EA artefacts, which are necessary to facilitate IT Management include applications, software components and hardware assets as well as the project portfolio. In this context EA is understood as a complementing approach to CMDB or IT Project Management.

Table 7. Factor 3—Support of Business Operations

Item 3.1	Security Management
Item 3.2	Sourcing and Business Networking
Item 3.5	Compliance Management

Factor 3 describes EA applications, which support daily business operations (Table 7). In contrast to factor 1 *Support of Business Strategy Development* EA is more concerned with maintaining the conditions and quality attributes the business requires to carry out its operations and only to a less extent with long term planning. By e. g. analysing business process roles, their correct embedment within the authorization structure of corresponding applications, EA ensures that the organization's identity management is aligned and consistent with business requirements. To support sourcing decisions and maintain SLA, EA provides transparency regarding process interfaces. This enables the organization to analyze whether such interfaces are compliant to a service provider. In this scenario required artefacts range from business processes, interfaces, and SLA to business networking partners.

5 Summary and Future Work

Based on the discussion in situational method engineering and the current EA state-of-the-art, this paper suggests differentiating contingency factors of EA. The results of the exploratory analysis confirm the assumption that there is no overall approach to adapt to EA in practice, but to distinguish between three EA realization approaches. They represent three different approaches on how to grasp EA in terms of its determining factors. The exploratory analysis (Fig. 2) shows that *adoption of advanced architectural design paradigms and modelling capabilities,* and *organizational penetration of EA* are significant factors to discriminate between different EA approaches in practice. The fact that EA (as opposed to IT architecture) is a pretty novel topic is addressed by an analysis of possible EA applications. The presented contingency factors, resulting in different realization approaches and application scenarios provide a basis on which EA methods can be adapted to a specific situation.

A possibility to consolidate and validate the findings of the analysis at hand is to build EA methods employing these contingency factors, respectively the EA approaches and application scenarios, and evaluate these methods in real life case studies. This will help to further enhance the construction of methods for an effective EA management, where methods specifically fit to the situations in which they are applied.

Although we do not interpret the identified EA realization approaches as levels of EA maturity, we expect the members of each cluster to further develop their EA. This is an important starting point for further research activities, where the exploration, description, and in particular the methodical support of such transformation or development paths have to be covered.

References

1. Aier, S., Riege, C., Winter, R.: Classification of Enterprise Architecture Scenarios – An Exploratory Analysis. Enterprise Modelling and Information Systems Architectures 3(1), 14–23 (2008)
2. Arbab, F., de Boer, F., Bonsangue, M., Lankhorst, M., Proper, E., van der Torre, L.: Integrating Architectural Models. Symbolic, Semantic and Subjective Models in Enterprise Architecture. Enterprise Modelling And Information System Architectures 2(1), 40–56 (2007)

3. Bernard, S.A.: An Introduction to Enterprise Architecture, 2nd edn. Authorhouse, Bloomington (2005)
4. Buckl, S., Ernst, A.M., Lankes, J., Matthes, F., Schweda, C.M., Wittenburg, A.: Generating Visualizations of Enterprise Architectures using Model Transformations. Enterprise Modelling and Information Systems Architectures 2(2), 3–13 (2007)
5. Cattell, R.B.: Factor Analysis: An Introduction and Manual for the Psychologist and Social Scientist. Harper and Row, New York (1952)
6. Dziuban, C.D., Shirkey, E.C.: When is a correlation matrix appropriate for factor analysis? Psychological Bulletin 81(6), 358–361 (1974)
7. Fischer, R., Aier, S., Winter, R.: A Federated Approach to Enterprise Architecture Model Maintenance. Enterprise Modelling and Information Systems Architectures 2(2), 14–22 (2007)
8. Frank, U.: Multi-Perspective Enterprise Modeling (MEMO) – Conceptual Framework and Modeling Languages. In: Proceedings of the Hawaii International Conference on System Sciences (HICSS-35) (2002)
9. Gordon, A.D.: Hierarchical Classification. In: Arabie, P., Hubert, L.J., De Soete, G. (eds.) Clustering and Classification, pp. 65–121. World Scientific Publishing, River Edge (1996)
10. Hair Jr., J.F., Black, B., Babin, B.: Multivariate Data Analysis, 6th edn. Prentice Hall, Englewood Cliffs (2006)
11. Härdle, W., Simar, L.: Applied Multivariate Statistical Analysis. Springer, Berlin (2003)
12. Harmsen, A.F., Brinkkemper, S., Oei, H.: Situational Method Engineering for Information System Project Approaches. In: Proceedings of the IFIP 8.1 Working Conference on Methods and Associated Tools for the Information Systems Life Cycle, pp. 169–194. North-Holland, Amsterdam (1994)
13. Huisman, M., Iivari, J.: The individual deployment of systems development methodologies. In: Pidduck, A.B., Mylopoulos, J., Woo, C.C., Ozsu, M.T. (eds.) CAiSE 2002. LNCS, vol. 2348, pp. 134–150. Springer, Heidelberg (2002)
14. Johnson, P., Ekstedt, M.: Enterprise Architecture – Models and Analyses for Information Systems Decision Making. Studentlitteratur, Pozkal (2007)
15. Jonkers, H., Lankhorst, M., van Buuren, R., Hoppenbrouwers, S., Bonsangue, M., van der Torre, L.: Concepts for Modelling Enterprise Architectures. International Journal of Cooperative Information Systems 13(3), 257–287 (2004)
16. Kaiser, H.F., Rice, J.: Little Jiffy, Mark IV. Educational and Psychological Measurement 34(1), 111–117 (1974)
17. Kettinger, W.J., Teng, J.T.C., Guha, S.: Business Process Change: A Study of Methodologies, Techniques, and Tools. MISQ 21(1), 55–80 (1997)
18. Kumar, K., Welke, R.J.: Methodology Engineering – A Proposal for Situation-specific Methodology Construction. In: Cotterman, W., Senn, J.A. (eds.) Challenges and Strategies for Research in Systems Development, pp. 257–269. John Wiley & Sons, New York (1992)
19. Lankhorst, M.: Enterprise Architecture at Work: Modelling, Communication and Analysis. Springer, Berlin (2005)
20. Niemann, K.D.: From Enterprise Architecture to IT Governance. Elements of Effective IT Management. Vieweg, Wiesbaden (2006)
21. Pereira, C.M., Sousa, P.: A Method to Define an Enterprise Architecture using the Zachman Framework. In: Proceedings of the 2004 ACM Symposium On Applied Computing, pp. 1366–1371. ACM Press, New York (2004)
22. Ross, J.W., Weill, P., Robertson, D.C.: Enterprise Architecture as Strategy. Creating a Foundation for Business Execution. Harvard Business School Press, Boston (2006)

23. Rummel, R.J.: Applied Factor Analysis. Northwestern University Press, Chicago (1970)
24. Schekkerman, J.: How to Survive in the Jungle of Enterprise Architecture Frameworks: Creating or Choosing an Enterprise Architecture Framework, 2nd edn. Trafford Publishing, Victoria (2004)
25. Schelp, J., Stutz, M.: A Balanced Scorecard Approach to Measure the Value of Enterprise Architecture. Journal of Enterprise Architecture 3(4), 8–14 (2007)
26. van Slooten, K., Hodes, B.: Characterizing IS Development Projects. In: Proceedings of the IFIP TC8, WG8.1/8.2 Working Conference on Method Engineering, pp. 29–44. Springer, Berlin (1996)
27. Spewak, S.H., Hill, S.C.: Enterprise Architecture Planning – Developing a Blueprint for Data, Applications and Technology. John Wiley & Sons, New York (1993)
28. The Open Group, The Open Group Architecture Framework TOGAF – 2007 Edition (Incorporating 8.1.1). Van Haren, Zaltbommel (2007)
29. Theuerkorn, F.: Lightweight Enterprise Architectures. Auerbach Publishers, Boca Raton (2004)
30. Veasey, P.W.: Use of enterprise architectures in managing strategic change. Business Process Management Journal 7(5), 420–436 (2001)
31. Wegmann, A.: The Systemic Enterprise Architecture Methodology (SEAM) – Business and IT Alignment for Competiveness, École Poytechnique Fédérale de Lausanne (2002)
32. Winter, R., Bucher, T., Fischer, R., Kurpjuweit, S.: Analysis and Application Scenarios of Enterprise Architecture – An Exploratory Study. Journal of Enterprise Architecture 3(3), 33–43 (2007)
33. Winter, R., Fischer, R.: Essential Layers, Artifacts, and Dependencies of Enterprise Architecture. Journal of Enterprise Architecture 3(2), 7–18 (2007)
34. Ylimäki, T., Halttunen, V.: Method engineering in practice: A case of applying the Zachman framework in the context of small enterprise architecture oriented projects. Information Knowledge Systems Management 5(3), 189–209 (2006)
35. Zachman, J.A.: A Framework for Information Systems Architecture. IBM Systems Journal 26(3), 276–292 (1987)

Towards a Common Terminology in the Discipline of Enterprise Architecture

Marten Schöenherr

Deutsche Telekom Laboratories (T-Labs), Technical University of Berlin, Germany
marten.schoenherr@telekom.de

Abstract. This paper presents a literature analysis considering 126 references to support a common terminology in the discipline of Enterprise Architecture (EA). In a first step, it surveys EA-Drivers, addressed architectural layers and the differentiation of contributions focusing on architectural descriptions and architectural development.

Keywords: Enterprise Architecture, Terminology, Drivers, Architecture Layers.

1 Motivation

An increasing number of publications refer to the term or even the discipline of Enterprise Architecture (EA). Some authors point out that there is no common understanding of the term [28, 66, 82]. Especially in science, an un-reflected usage of buzzwords hinders experts to discuss relevant issues in an appropriate way. This contribution will deliver dimensions to differentiate approaches based on a literature analysis. The following sections will show results of a neutral comparative survey to find possible dimensions to describe the focus of an EA-contribution. The paper starts with some basic facts as timeline of the chosen literature, the authors backgrounds, main issues focused in the contributions and the handling of definitions and terminology. Afterwards three dimensions are described: EA-Drivers, addressed architectural layers and the differentiation between architectural descriptions and architectural development. It closes with a conclusion.

2 Research Methodology

The following thoughts are based on a literature analysis using 126 references from academic and pragmatic sources. This includes research publications (journals, conference proceedings), books and websites. The considered references have been surveyed via the internet.[1] Primer search-criteria have been the explicit usage of the term *Enterprise Architecture*. After a brief survey of the identified contributions, many non-research references needed to be excluded because they are marketing material. Furthermore, a next search covering aspects as Business-IT-Alignment,

[1] Google Scholar, SpringerLink, Google Web Search.

G. Feuerlicht and W. Lamersdorf (Eds.): ICSOC 2008, LNCS 5472, pp. 400–413, 2009.

IT-Architecture Management, Enterprise Modeling and Interoperability was performed to include work that is relevant but not explicitly labeled as EA-literature. After choosing the contributions to be considered the paper differentiates simple facts as time of publication, background of the author and how the authors deal with terminology and definitions of EA which are collected by reading the papers and summarizing the content without interpretation of implicit issues. In a second step the main points of interest as drivers for EA-considerations, addressed architectural layers and whether the focus is more on architectural description or architectural development will be surveyed with a little more effort. First, the contributions that explicitly mention an aspect (e.g. driver) have been evaluated and as a result, a scale of the dimension has been derived (e.g. internal vs. external drivers). Using the result from the first step the rest of the contributions were interpreted in the context of the deeper understanding of the dimension.

3 Literature Analysis

This chapter starts with the basic facts mentioned in the research methodology followed by examinations of EA-drivers, addressed architectural layers and the differentiation between aspects of architectural description and development.

3.1 Basic Facts, Distributions and Correlations

Since 2003, more and more authors are using the term EA explicitly in their publications. Most of the newer contributions are coming from an academic background. Especially after 2005, a lot of consultancies and IT-companies are adopting their products and strategies to an extended architectural understanding hence Enterprise Architecture. Based on the data sample there is no significant correlation between the time of a publication and the background of the author(s). There is a notion that papers from non-academic authors published before 2000 often fulfill at least basic academic or even scientific requirements. Considering that after 2004-2005 a lot of companies started to use the term EA and since then have connected it somehow to their products and strategies a huge amount of superficial marketing material has been distributed. Table 1 and Table 2 show the distribution of all references.

Considering the maturity and the focus of the contributions there is no core topic or even a theory in the discipline of EA. Almost half of the approaches discussed in the papers are still coming with a low maturity level (Concept Phase) in the context of readiness to be used in an organization. Only a third of the authors are delivering some kind of best practice (Implementation/Adoption). Differentiating the focus of EA-authors there are two specific topics only (EA-Frameworks and Enterprise Modeling) the majority is dealing with rather general aspects. A correlation between early maturity levels (Concept Phase) and modeling approaches is existing, mainly delivered by researchers. Many authors from the Public Sector are addressing EA-Frameworks in a descriptive way, but on an academic standard (e.g. FEAF, DoDAF etc.) – see Table 3 and 4.

Table 1. Publication timeline

Year	Contribution	#
1987	[123]	1
1988		0
1989		0
1990		0
1991		0
1992	[103]	1
1993		0
1994	[75], [88]	2
1995	[40]	1
1996	[30], [62], [65], [71], [77], [121], [124]	7
1997	[104]	1
1998	[10]	1
1999	[3], [8], [9], [13], [16], [125]	6
2000	[4], [19], [24], [83]	4
2001	[5], [20], [31], [35], [39], [81], [112], [120]	8
2002	[6], [23], [28], [101], [106], [108], [119]	7
2003	[7], [15], [25], [33], [44], [46], [48], [49], [55], [56], [64], [74], [84], [89], [91], [92], [93], [109], [114], [117]	20
2004	[12], [17], [26], [29], [45], [51], [54], [59], [63], [67], [70], [72], [73], [79], [82], [94], [95], [97], [105], [107], [113], [115]	22
2005	[2], [47], [57], [61], [69], [78], [87], [98], [102], [116]	10
2006	[1], [14], [27], [42], [43], [52], [66], [76], [85], [86], [90], [99], [111], [126]	14
2007	[11], [18], [21], [22], [34], [36], [37], [38], [41], [50], [53], [58], [60], [68], 80], [96], [100], [110], [118], [122]	20
2008	[32]	1

Table 2. Author´s background

Origin	Contribution	#
Academics	[1], [2], [4], [10], [11], [13], [15], [17], [18], [21], [22], [23], [24], [26], [27], [29], [30], [31], [32], [37], [38], [39], [40], [42], [44], [46], [47], [51], [52], [53], [54], [55], [56], [60], [62], [63], [64], [65], [68], [69], [73], [79], [80], [83], [88], [89], [90], [92], [93], [95], [96], [97], [98], [101], [102], [104], [105], [106], [107], [108], [110], [111], [114], [115], [117], [118], [120], [121], [122], [126]	70
Public Sector	[5], [6], [7], [8], [9], [12], [19], [57]	8
Consulting-Company	[16], [20], [28], [33], [35], [36], [48], [50], [58], [59], [67], [70], [71], [75], [76], [78], [82], [85], [87], [91], [94], [99], [109], [112], [113], [124], [125]	27
IT-Company	[3], [34], [41], [49], [66], [81], [103], [119], [123]	9
various	[14], [25], [43], [45], [61], [72], [74], [77], [84], [86], [100], [116]	12

Table 3. Maturity level of contributions focused issues

Maturity	Contribution	#
Concept Phase	[4], [5], [17], [18], [21], [22], [23], [24], [26], [27], [29], [30], [32], [37], [38], [40], [42], [43], [44], [47], [50], [51], [52], [53], [54], [55], [56], [57], [59], [60], [63], [65], [68], [69], [71], [72], [73], [77], [79], [83], [85], [88], [89], [91], [92], [97], [98], [102], [103], [106], [107], [111], [114], [115], [117], [118], [120], [123], [126]	59
Towards some kind of Product	[1], [3], [7], [12], [20], [28], [31], [39], [48], [64], [66], [67], [75], [76], [78], [84], [86], [87], [93], [94], [95], [96], [99], [108], [109], [121]	26
Implementation/Adoption	[2], [6], [8], [9], [10], [11], [13], [14], [15], [16], [19], [25], [33], [34], [35], [36], [41], [45], [46], [49], [58], [61], [62], [70], [74], [80], [81], [82], [90], [100], [101], [104], [105], [110], [112], [113], [116], [119], [122], [124], [125]	41

While the first relevant publications go back to the End of the 1980s and the topic has been heavily discussed for the last ten years, only a third deals with validated best practice. Only 6% of the considered publications do give its own definition of the term EA and at the same time differentiate it to others by referring to their definitions (see Table 5). A small percentage is defining the term *Enterprise* because the term *Architecture* is not being used often in the context of domains as Managerial and Organizational Science (but with a long history in Computer Science- see Table 6).

Table 4. Focused issue of contribution

Adressed Issues	Contribution	#
Overview on EA	[4], [5], [8], [15], [23], [28], [29], [32], [33], [34], [44], [47], [49], [50], [61], [66], [71], [72], [74], [77], [79], [84], [87], [88], [93], [98], [99], [105], [107], [108], [114], [115], [120], [125]	34
Best Practice	[2], [6], [9], [10], [11], [12], [13], [14], [16], [19], [20], [25], [26], [27], [31], [35], [37], [39], [41], [43], [45], [46], [48], [52], [54], [58], [63], [64], [68], [70], [75], [78], [80], [81], [89], [90], [94], [95], [100], [101], [106], [109], [110], [112], [116], [122]	46
EA-Frameworks	[1], [24], [36], [38], [40], [60], [67], [73], [76], [82], [83], [92], [97], [103], [104], [113], [121], [123], [124], [126]	20
Enterprise Modeling	[3], [7], [17], [21], [42], [51], [53], [55], [56], [59], [65], [69], [102], [111], [117], [119]	16
various	[18], [22], [30], [57], [62], [85], [86], [91], [96], [118]	10

Table 5. Proprietary EA-definitions and references to other authors definitions

EA-Definition	Contribution	#
Proprietary Definition, no further References	[4], [5], [6], [8], [9], [15], [16], [20], [23], [25], [28], [30], [33], [39], [41], [43], [45], [47], [48], [53], [65], [66], [67], [68], [72], [74], [76], [77], [78], [80], [81], [84], [86], [88], [89], [90], [94], [95], [101], [105], [119], [124]	42
Proprietary Definition in the context of further defining References	[1], [44], [58], [93], [99], [116], [126]	7
Definition by References, no proprietary Definition	[2], [14], [17], [19], [26], [32], [34], [37], [38], [42], [46], [49], [50], [51], [52], [54], [57], [60], [61], [63], [69], [82], [87], [92], [96], [97], [98], [102], [108], [113], [114], [115], [117]	33
No Definition at all	[3], [7], [10], [11], [12], [13], [18], [21], [22], [24], [27], [29], [31], [35], [36], [40], [55], [56], [59], [62], [64], [70], [71], [73], [75], [79], [83], [85], [91], [100], [103], [104], [106], [107], [109], [110], [111], [112], [118], [120], [121], [122], [123], [125]	44

Table 6. Including the term *Enterprise* to an extended architectural understanding

Defining the term *Enterprise*	Contribution	#
yes	[1], [3], [5], [9], [15], [44], [47], [65], [68], [79], [81], [88], [91], [93], [94], [108], [117], [119]	18
no	Rest of 126 considered References	108

50% of the authors are technology-driven. 36% are following a systemic approach towards a wider and integrated architectural understanding, which includes at least another architectural layer apart from the IT-Architecture. 14% tend to use a method-driven terminology. Some of the authors even combine the three main directions (see Table 7 and Table 8). To summarize the first section: there is a lack of theoretical foundation, stringent definitions or a common understanding within the authors, who publish in the context of EA. The majority of authors are publishing with a research background, they are technology-oriented and most of the introduced approaches are still in a concept phase and have not proven neither their real world value nor their feasibility.

Table 7. Term-Definitions main focus

Term-definitions main focus	Contribution	#
systemic	[1], [4], [8], [9], [25], [30], [66], [67], [72], [74], [76], [80], [81], [88], [90], [94], [95], [99], [105], [119], [126]	21
technology-driven	[1], [5], [8], [9], [20], [23], [28], [33], [39], [43], [44], [45], [48], [53], [58], [65], [68], [74], [78], [80], [84], [86], [89], [93], [101], [116], [119], [124], [126]	29
method-driven	[1], [6], [15], [16], [41], [47], [77], [116]	8

Table 8. Combinations of focused issues

Combinations	Contribution	#
systemic	[4], [25], [30], [66], [67], [72], [76], [81], [88], [90], [94], [95], [99], [105]	14
technology-driven	[5], [20], [23], [28], [33], [39], [43], [44], [45], [48], [53], [58], [65], [68], [78], [84], [86], [89], [93], [101], [124]	21
method-driven	[6], [15], [16], [41], [47], [77]	6
systemic & techn.-driven	[8], [9], [74], [80], [119], [126]	6
systemic & method-driven		0
techn.- & method-driven	[116]	1
sys. & techn.- & meth.-driven	[1]	1

3.2 Drivers for Enterprise Architecture Approaches

A central part of a common understanding could be the reasons why organizations are supposed to gain advantages from EA-approaches. Therefore, the drivers mentioned in the considered contributions have been surveyed. Just a small minority of authors are discussing drivers, why organizations are interested in EA. They differentiate internal and external drivers (see details in Table 9). In the category of internal drivers *Business-IT-Alignment* (by far) and *Cost-Reduction* are the most common entries. External drivers are legal requirements that push organizations to improve their Business-IT-Alignment.

Table 9. Distribution on internal and external EA-Drivers

Driver		Contribution	#
Internal	Business-IT Alignment	[5], [18], [42], [49], [57], [60], [61], [87], [115], [118], [122]	11
	Cost Reduction	[19] , [20] , [84], [86]	4
	Standardization/Consolidation	[84], [86]	2
	Management/Governance	[32], [84]	2
	various	[5], [14], [37], [46], [57], [84]	6
External	Clinger-Cohen Act	[5], [47], [18], [57], [58], [61], [99], [119]	8
	Sarbanes-Oxley Act	[34], [37], [58], [61], [78], [118]	6
	Basel II	[37], [58], [61], [118]	4
	Solvency II	[37], [58], [78]	3
	various	[5], [34], [42], [58]	4

3.3 Architectural Layers Addressed in EA-Contributions

An extended architectural understanding should consider elements apart from IT-Architectures. The authors are naming their layers on many different ways. The used categorization (Strategy, Organization, Information, Integration/Interoperability, Application/Appl.-Landscape and Infrastructure) has been derived considering all contributions that explicitly name an architectural layer concept and their generalization. Just counting the layers described, more papers deal with non-technical layers (Strategy, Organization and Information) than technical layers (the others). More than half of the authors are not addressing some kind of architectural layer or just one single layer

hence not even half of the authors are dealing with two or more architectural layers, which would be expected in the context of an EA-approach (see Table 10 and 11).

Differentiating the focused layer within the contributions that address one layer only, by far most of the authors are dealing with organizational issues. The architectural layer *Organization* can be divided into business processes, organizational structures and a mixture of both. More than half of the authors speak about business processes in the context of Organization. Hence a majority addresses business process aspects. Second common is the issue *Applications* and/or *Application Landscape* (see Table 12 and 13).

Table 10. Distribution of addressed architectural layers

EA-Layer	Contribution	#
Strategy	[14], [15], [16], [19], [28], [35], [53], [54], [58], [70], [78], [84], [89], [90], [112], [116]	16
Organization	[3], [6], [7], [11], [14], [15], [17], [18], [20], [25], [26], [32], [39], [40], [42], [44], [48], [49], [50], [53], [55], [58], [59], [60], [61], [63], [65], [68], [69], [73], [74], [75], [78], [82], [83], [87], [90], [91], [93], [94], [96], [98], [99], [101], [105], [110], [115], [116], [118], [120], [121], [122]	52
Information	[6], [7], [20], [34], [39], [46], [48], [49], [50], [53], [56], [59], [60], [61], [62], [68], [69], [73], [74], [82], [83], [94], [102], [106], [115], [117], [121], [122], [123]	29
Integration	[2], [20], [26], [31], [46], [63], [101], [113]	8
Applications/Appl.-Landscape	[4], [10], [16], [21], [24], [25], [26], [27], [33], [40], [43], [44], [45], [48], [49], [51], [56], [58], [61], [70], [74], [79], [87], [93], [98], [104], [105], [106], [111], [113], [114], [122], [124]	33
Infrastructure	[10], [12], [16], [25], [33], [34], [43], [44], [62], [75], [84], [87], [104], [108], [109], [110], [123], [124]	18

Table 11. Distribution on the overall number of considered layers

# of adressed Layers	Contribution	#
0	[1], [5], [8], [9], [13], [22], [23], [29], [30], [36], [37], [38], [41], [47], [52], [57], [64], [66], [67], [71], [72], [76], [77], [80], [81], [85], [86], [88], [92], [95], [97], [100], [103], [107], [119], [125], [126]	37
1	[2], [3], [4], [11], [12], [17], [18], [19], [21], [24], [27], [28], [31], [32], [35], [42], [45], [51], [54], [55], [65], [79], [89], [91], [96], [99], [102], [108], [109], [111], [112], [114], [117], [118], [120]	35
2	[6], [7], [10], [14], [15], [33], [34], [39], [40], [43], [46], [50], [56], [59], [60], [62], [63], [68], [69], [70], [73], [75], [78], [82], [83], [84], [90], [93], [94], [98], [101], [104], [105], [106], [110], [113], [115], [116], [121], [123], [124]	41
3	[16], [20], [25], [26], [44], [48], [49], [53], [58], [61], [74], [87], [122]	13

Table 12. Distribution of addressed architectural layers (one layer addressed only)

Single layer adressed	Contribution	#
Strategy	[19], [28], [35], [54], [89], [112]	6
Organization	[3], [11], [17], [18], [32], [42], [55], [65], [91], [96], [99], [118], [120]	13
Information	[102], [117]	2
Integration/Interoperability	[2], [31]	2
Application/Appl.-Landscape	[4], [21], [24], [27], [45], [51], [79], [111], [114]	9
Infrastructure	[12], [108], [109]	3

Table 13. Distribution of architectural layer *Organization*

Organizational issues	Contribution	#
Organizational Structures	[68], [91], [94], [122]	4
Business Processes	[19], [28], [35], [54], [89], [112] , [3], [7], [11], [17], [20], [26], [39], [40], [42], [48], [49], [50], [55], [59], [63], [65], [69], [78], [87], [93], [96], [98], [99], [105], [115], [116], [120]	27
Structures & Processes	[6], [14], [15], [18], [25], [32], [44], [53], [58], [60], [61], [73], [74], [75], [82], [83], [90], [101], [110], [118], [121]	21

After a deeper look on the combined architectural layers (when two layers are addressed), by far most approaches do combine organizational issues with the information layer. The information layer includes all aspects towards business and technology matter of information systems as well as data models.

Summarizing the surveyed architectural layers, too few authors are addressing hence integrating multiple layers. When multiple layers are focused on, the most common constellation combines business process artifacts with more conceptional issues of information systems. Hence, the understanding of Business-IT-Alignment does not consider enterprise strategy and technical details from the mostly technical architectural layers (Integration/Interoperability, Applications/Application Landscape and Infrastructure).

Considering similar scientific disciplines, neither Managerial and Behavioral Science nor Computer Science or even Electrical Engineering is suitable for that focus. The only domain that discusses this combination (incl. the topic Enterprise Modeling) is Information Systems Research (ISR).

Furthermore there is no significant accumulation to identify another bundle of topics, hence the EA-Community does not have a common targeted issue.

3.4 Architecture Description vs. Architecture Development

The last considered differentiation between the chosen contributions is based on the ISO 15704 [129]. The norm defines two types of addressing an extended architectural understanding. Type 1 is summarizing all approaches that focus on aspects to describe the state of an AS IS and/or TO BE architecture (incl. static and dynamic issues). Type 2 extends the Type 1 considerations with methodologies how to develop an AS IS state towards a planned TO BE. Many of the EA-Frameworks can be called Type 2 approaches. Type 1 and Type 2 Architectures are complementary. Without the description of AS IS and TO BE, there is no such thing as a meaningful methodology to derive a better TO BE Architecture. ISO 15704 does not define the scope (or layer) of an architectural consideration; therefore, it can be used in the context of EA. It is explicitly addressing Enterprise Reference Architectures.

Within the surveyed contributions there is no majority towards some kind of architectural type. Hence, the authors deal with documenting issues as well as with methodologies (or the combination of both).

Table 14. Distribution architectural description vs. architectural development

Type of Architectural Consideration	Contribution	#
Typ 1	[2], [3], [7], [10], [13], [17], [21], [26], [30], [39], [42], [49], [51], [53], [54], [55], [56], [59], [63], [65], [87], [94], [102], [105], [106], [108], [111], [113], [114], [119], [120]	31
Typ 2	[1], [5], [8], [9], [12], [14], [18], [19], [28], [32], [36], [45], [47], [67], [69], [73], [76], [79], [80], [82], [83], [84], [90], [92], [93], [95], [97], [101], [107], [110], [116], [117], [121], [122], [126]	35
Typ 1+2	[6], [11], [15], [16], [24], [25], [34], [37], [48], [50], [58], [66], [74], [81], [86], [100], [103], [104], [115], [123], [124]	21
No Typ	[4], [20], [22], [23], [27], [29], [31], [33], [35], [38], [40], [41], [43], [44], [46], [52], [57], [60], [61], [62], [64], [68], [70], [71], [72], [75], [77], [78], [85], [88], [89], [91], [96], [98], [99], [109], [112], [118], [125]	39

It is possible to differentiate the contributions along these criteria very well, but except from [15] the explicit usage is not seen in any of the contributions. It seems to be quite a useful differentiation because it is simple and disjunctive.

4 Conclusion

As stated in the introduction: this is not about yet another EA-Definition. It is about a lack of a common terminology. It is not possible to consider all EA-Publications. Many of the results are based on interpretation of implicit statements. That makes it difficult to call this literature analysis scientific.

Nevertheless, there is no doubt about the horrible mess looking at the usage of the term *Enterprise Architecture*! The only way to improve the situation is to start thinking about a common structure, developing a core theory and please: define and differentiate your aims, methods and addressed issues. This paper would like to give some first orientation and maybe start a discussion in the EA-community.

Another result of the survey is the still blur but developing picture, that authors can be differentiated into descriptive and descriptive-normative positions. Descriptive approaches see EA as a result of a planning process. The descriptive-normative authors of contributions consider the planning process as an integrative part of an EA-Approach.

References

1. TOGAF Version 8.1.1, `http://www.opengroup.org/architecture/togaf8-doc/arch`
2. Anaya, V., Ortiz, A.: How enterprise architectures can support integration. In: Proceedings of the First international workshop on Interoperability of heterogeneous information systems. ACM, Bremen (2005)
3. Anonymous, Enterprise Modeling: Aligning Business and IT, Popkin Software (1999)
4. Anonymous, IEEE Recommended Practice for Architectural Description of Software-Intensive Systems (IEEE Std 1471-2000), The Institute of Electrical and Electronics Engineers, Inc. (2000)
5. Anonymous, A Practical Guide to Federal Enterprise Architecture. Chief Information Officer Council (2001)
6. Anonymous, Enterprise Architecture (2001)
7. Armour, F., et al.: A UML-Driven Enterprise Architecture Case Study. In: Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS 2003). IEEE Computer Society, Los Alamitos (2003)
8. Armour, F.J., Kaisler, S.H., Liu, S.Y.: A Big-Picture Look at Enterprise Architectures. IT Professional 1(1), 35–42 (1999)
9. Armour, F.J., Kaisler, S.H., Liu, S.Y.: Building an Enterprise Architecture Step by Step. IT Professional 1(4), 31–39 (1999)
10. Bahrami, A., Sadowski, D., Bahrami, S.: Enterprise architecture for business process simulation. In: Proceedings of the 30th conference on Winter simulation. IEEE Computer Society Press, Washington (1998)

11. Barjis, J., Barjis, I.: Business Process Modeling as a Blueprint for Enterprise Architecture. In: Saha, P. (ed.) Handbook Of Enterprise Systems Architecture In Practice, pp. 114–128. Information Science Reference, London (2007)
12. Bass, T., Mabry, R.: Enterprise Architecture Reference Models: A Shared Vision for Service-Oriented Architectures. In: Proceedings of the to IEEE MILCOM 2004 (2004) (for submission)
13. Belle, J.-P.V.: Moving Towards Generic Enterprise Information Models: From Paciolo to Cyc. In: Proceedings of the Australian Conference on Information Systems (1999)
14. Berg, M.v.d., Steenbergen, M.v.: Building An Enterprise Architecture Practice. Springer, Heidelberg (2006)
15. Bernus, P., Nemes, L., Schmidt, G.: Handbook on Enterprise Architecture. Springer, Heidelberg (2003)
16. Boar, B.H.: Constructing blueprints for Enterprise IT Architectures. John Wiley & Sons, Inc., Chichester (1999)
17. Boer, F.S.d., et al.: A Logical Viewpoint on Architectures. In: Proceedings of the Enterprise Distributed Object Computing Conference, Eighth IEEE International. IEEE Computer Society, Los Alamitos (2004)
18. Bommel, P.v., et al.: Architecture principles – A regulative perspective on enterprise architecture. In: Proceedings of the 2nd International Workshop on Enterprise Modelling and Information Systems Architectures. Gesellschaft für Informatik, Köllen, St. Goar (2007)
19. Brown, D.P.: Enterprise Architecture for DoD (Department of Defense) Acquisition. Acquisition Review Quarterly, 121–130 (2000)
20. Buchanan, R.D., Soley, R.M.: Aligning Enterprise Architecture and IT Investments with Corporate Goals, Meta Group (2002)
21. Buckl, S., et al.: Generating Visualizations of Enterprise Architectures Using Model Transfomations. In: Proceedings of the 2nd International Workshop on Enterprise Modelling and Information Systems Architectures. Gesellschaft für Informatik, St. Goar (2007)
22. Cane, S., McCarthy, R.: Measuring The Impact Of Enterprise Architecture, pp. 437–442 (2007)
23. Chung, H.M., McLeod, G.: Enterprise Architecture, Implementation, and Infrastructure Management. In: Proceedings of the 35th Hawaii International Conference on System Sciences (2002)
24. Dedene, G., Maes, R.: On the integration of the Unified Process Model in a framework for software architecture. In: Proceedings of the Landelijk Architectuur Congress (2000)
25. Dern, G.: Management von IT-Architekturen. Vieweg & Sohn Verlag, Firdr (2003)
26. Doest, H.t., Lankhorst, M.: Tool Support for Enterprise Architecture – A Vision (2006)
27. Dreyfus, D., Iyer, B.: Enterprise Architecture: A Social Network Perspective. In: Proceedings of the 39th Annual Hawaii International Conference on System Sciences, vol. 08. IEEE Computer Society, Los Alamitos (2006)
28. Drobik, A.: Enterprise Architecture: The Business Issues and Drivers. Gartner, Inc. (2002)
29. Ekstedt, M.: Enterprise Architecture as a Means for IT Management (2004)
30. Ellis, W., et al.: Toward a Recommended Practice for Architectural Description. In: Proceedings of the 2nd IEEE International Conference on Engineering of Complex Computer Systems (ICECCS 1996). IEEE Computer Society, Los Alamitos (1996)
31. Emmerich, W., Ellmer, E., Fieglein, H.: TIGRA: an architectural style for enterprise application integration. In: Proceedings of the 23rd International Conference on Software Engineering. IEEE Computer Society, Toronto (2001)

32. Espinosa, J.A., Armour, F.: Geographically Distributed Enterprise Architecting: Towards a Theoretical Framework. In: Proceedings of the Proceedings of the 41st Annual Hawaii International Conference on System Sciences. IEEE Computer Society, Los Alamitos (2008)
33. Evernden, R., Evernden, E.: Third-generation information architecture. Commun. ACM 46(3), 95–98 (2003)
34. Feurer, S.: Enterprise Architecture – An Overview, SAP AG (2007)
35. Finkelstein, C.: Enterprise Portal Succes: Enterprise Architecture. DM Review Enterprise Column (2001)
36. Finkelstein, C.: Introduction to Enterprise Architecture (2007)
37. Fischer, R., Aier, S., Winter, R.: A Federated Approach to Enterprise Architecture Model Maintenance. In: Proceedings of the 2nd International Workshop on Enterprise Modelling and Information Systems Architectures. Gesellschaft für Informatik, Köllen, St. Goar (2007)
38. Foorthuis, R.M., Brinkkemper, S.: A Framework for Project Architecture in the Context of Enterprise Architecture. In: Proceedings of the 2nd Workshop on Trends in Enterprise Architecture Research 2007 (2007)
39. France, R.B., Ghosh, S., Turk, D.E.: Towards a Model-Driven Approach to Reuse. In: Proceedings of the OOIS 2001, Calgary, Canada (2001)
40. Fraser, J., Tate, A.: The Enterprise Tool Set An Open Enterprise Architecture. In: Proceedings of the The 1995 International Joint Conference on AI (1995)
41. Gerber, S., Meyer, U., Richert, C.: EA Model as central part of the transformation into a more flexible and powerful organisation. In: Proceedings of the 2nd International Workshop on Enterprise Modelling and Information Systems Architectures. Gesellschaft für Informatik, St. Goar (2007)
42. Gils, B.v., Vojevodina, D.e.: The effects of exceptions on enterprise architecture. Radboud University Nijmegen, Nijmegen (2006)
43. Greefhorst, D.: Service Oriented Enterprise Architecture. In: Proceedings of the 2nd Workshop on Landelijk Architectuur Congres 2006 (2006)
44. Gronau, N.: Wandlungsfähige Informationssystemarchitekturen - Nachhaltigkeit bei organisatorischem Wandel. GITO-Verlag, Berlin (2003)
45. Hafner, M., Schelp, J., Winter, R.: Architekturmanagement als Basis effizienter und effektiver Produktion von IT-Services (2004)
46. Hamilton, J.A., Catania, M.G.A.: A Practical Application of Enterprise Architecture for Interoperability. In: Proceedings on The 2003 International Conference on Information Systems and Engineering (ISE 2003), Quebec, Canada (2003)
47. Harmon, K.: The "Systems" Nature of Enterprise Architecture. In: Proceedings of the 2005 International Conference on Systems, Man, and Cybernetics (2005)
48. Harmon, P.: Business Process Trends: Developing an Enterprise Architecture. Popkin Software (2003)
49. Hayes, H.: Demystifying Enterprise Architecture (2003)
50. Iacob, M.-E., et al.: Capturing Architecture for the Adaptive Enterprise (2007)
51. Iacob, M.-E., Leeuwen, D.v.: View Visualization For Enterprise Architecture. In: Proceedings of the 6th International Conference on Enterprise Information Systems (2004)
52. Janssen, M., Kuk, G.: A Complex Adaptive System Perspective of Enterprise Architecture in Electronic Government. In: Proceedings of the 39th Annual Hawaii International Conference on System Sciences, vol. 04. IEEE Computer Society, Los Alamitos (2006)
53. Johnson, P., Ekstedt, M.: Enterprise Architecture - Models and Analyses for Information Systems Decision Making (2007)

54. Johnson, P., et al.: Proceedings of the Using Enterprise Architecture For CIO Decision-Making: On The Importance Of Theory (2004)
55. Jonkers, H., et al.: Towards a Language for Coherent Enterprise Architecture Descriptions. In: Proceedings of the 7th International Conference on Enterprise Distributed Object Computing. IEEE Computer Society, Los Alamitos (2003)
56. Jonkers, H., et al.: Concepts for Modelling Enterprise Architectures. In: Proceedings of the Landelijk Architectuur Congres, Nieuwegein, The Netherlands (2003)
57. Kaisler, S.H., Armour, F., Valivullah, M.: Enterprise Architecting: Critical Problems. In: Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS 2005). IEEE Computer Society, Los Alamitos (2005)
58. Keller, W.: IT-Unternehmensarchitektur. dpunkt Verlag (2007)
59. Khoury, G.R., Simoff, S.J.: Enterprise architecture modelling using elastic metaphors. In: Proceedings of the First Asian-Pacific conference on Conceptual modelling, vol. 31. Australian Computer Society, Inc., Dunedin (2004)
60. Kourdi, M.E., Shah, H., Atkins, A.: A Proposed Framework for Knowledge Discovery in Enterprise Architecture. In: Proceedings of the 2nd Workshop on Trends in Enterprise Architecture Research 2007 (2007)
61. Lankhorst, M., et al.: Enterprise Architecture at Work: Modelling, Communication, and Analysis. Springer, Heidelberg (2005)
62. Laudato, N.C., DeSantis, D.J.: Managing the Implementation of an Enterprise-wide Information Architecture. In: Proceedings of the Cause annual conference (1996)
63. Leeuwen, D.v., Doest, H.t., Lankhorst, M.: A Tool Integration Workbench For Enterprise Architecture. In: Proceedings of the 6th International Conference on Enterprise Information Systems, Porto, Portugal (2004)
64. Leganza, G.: Project Governance and Enterprise Architecture Go Hand in Hand. Forrester Research, Inc. (2003)
65. Liles, D.H., Presley, A.R.: Enterprise modeling within an enterprise engineering framework. In: Proceedings of the 28th conference on Winter simulation. IEEE Computer Society, Coronado (1996)
66. Lillehagen, F., Karlsen, D.: Enterprise Architectures – Survey of Practices and Initiatives. In: Proceedings of the First International Conference on Interoperability of Enterprise Software and Applications, Geneva, Switzerland (2006)
67. Macaulay, A.: Enterprise Architecture Design and the Integrated Architecture Framework. Microsoft Architect Journal, 4–9 (2004)
68. Magalhaes, R., Zacarias, M., Tribole, J.: Making Sense of Enterprise Architectures as Tools of Organizational Self-Awareness (OSA). In: Proceedings of the 2nd Workshop on Trends in Enterprise Architecture Research 2007 (2007)
69. Magee, C., et al.: Successful Modelling of the Enterprise (2005)
70. Malan, R., Bredemeyer, D.: Architecture as Business Competency, Bredemeyer Consulting (2004)
71. Malhotra, Y.: Enterprise Architecture: An Overview (1996)
72. Malveau, R.: Bridging the Gap: Business and Software Architecture, Part 2 (2004)
73. Martin, R.A., Robertson, E.L., Springer, J.A.: Architectural Principles for Enterprise Frameworks: Guidance for Interoperability. In: Proceedings on the International Conference on Enterprise Integration Modelling and Technology 2004 (ICEIMT 2004), Toronto, Canada (2004)
74. McGovern, J., et al.: A Practical Guide To Enterprise Architecture (2003)
75. Melling, W.P.: Enterprise information architectures: they're finally changing. SIGMOD Rec. 23(2), 493–504 (1994)

76. Mulholland, A., Macaulay, A.L.: Architecture and the Integrated Architecture Framework (2006)
77. Nell, J.G.: Architectures and Frameworks (1996)
78. Niemann, K.D.: IT Governance and Enterprise Architecture - Impact of IT cost reduction on innovation power. The Journal of Enterprise Architecture 1, 31–40 (2005)
79. Nightingale, D.J., Rhodes, D.H.: Enterprise Systems Architecting: Emerging Art and Science within Engineering Systems. In: Proceedings of the ESD External Symposium (2004)
80. O'Neill, T., et al.: Managing Enterprise Architecture Change. In: Saha, P. (ed.) Handbook Of Enterprise Systems Architecture In Practice, pp. 192–205. Information Science Reference, London (2007)
81. Osvalds, G.: Definition of Enterprise Architecture-centric Models for the Systems Engineer. In: Proceedings of the 11th Annual International Symposium of the International Council on Systems Engineering (INCOSE), Melbourne, Australia (2001)
82. Pereira, C.M., Sousa, P.: A method to define an Enterprise Architecture using the Zachman Framework. In: Proceedings of the 2004 ACM symposium on Applied computing. ACM, Nicosia (2004)
83. Peristeras, V., Tarabanis, K.: Towards an enterprise architecture for public adminstration using a top-down approach. European Journal for Information Systems 9(4), 252–260 (2000)
84. Perks, C., Beveridge, T.: Guide To Enterprise IT Architecture. Springer, New York (2003)
85. Reekum, E.V.-V., et al.: An Instrument for Measuring the Quality of Enterprise Architecture Products. In: Proceedings of the 2nd Workshop on Landelijk Architectuur Congres 2006 (2006)
86. Rico, D.F.: A Framework for Measuring the ROI of Enterprise Architecture (2006)
87. Rohloff, M.: Enterprise Architecture – Framework and Methodology for the Design of Architectures in the Large. In: Proceedings of the 13th European Conference on Information Systems, Regensburg, Germany (2005)
88. Rood, M.A.: Enterprise Architecture: Definition, Content and Utility. In: Proceedings of the IEEE Third Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises. The MITRE Corporation (1994)
89. Ross, J.W.: Creating a Strategic IT Architecture Competency: Learning in Stages, Massachusetts Institute of Technology, USA (2003)
90. Ross, J.W., Weill, P., Robertson, D.: Enterprise Architecture As Strategy: Creating a Foundation for Business Execution. Harvard Business School Press (2006)
91. Rudawitz, D.: Why Enterprise Architecture Efforts Often Fall Short, Antevorte Consulting, LLC (2003)
92. Saha, P.: Analyzing The Open Group Architecture Framework from the GERAM Perspective (2003)
93. Schekkerman, J.: Enterprise Architecture Validation (2003)
94. Schekkerman, J.: Another View at Extended Enterprise Architecture Viewpoints. In: Proceedings of the Landelijk Architectuur Congres 2004 (2004)
95. Schekkerman, J.: Enterprise Architecture Scorecard (2004)
96. Schelp, J., Stutz, M.: A Balanced Scorecard Approach To Measure the Value of Enterprise Architecture. In: Proceedings of the 2nd Workshop on Trends in Enterprise Architecture Research 2007 (2007)
97. Schönherr, M.: Enterprise Architecture Frameworks, Enterprise Application Integration – Serviceorientierung und nachhaltige Architekturen, Gito, Berlin, pp. 3–48 (2004)

98. Schönherr, M., Aier, S.: Sustainable Enterprise Architecture - Central (EAI) vs. Dezentral (SOA) Approaches To Define Flexible Architectures (2005)
99. Sessions, R.: A Better Path To Enterprise Architectures (2006)
100. Sliva, R.: Enterprise Architecture by a Small Unit in a Federated Organization. In: Saha, P. (ed.) Handbook Of Enterprise Systems Architecture In Practice, pp. 320–330. Information Science Reference, London (2007)
101. Smith, D., et al.: A Roadmap for Enterprise Integration. In: Proceedings of the 10th International Workshop on Software Technology and Engineering Practice. IEEE Computer Society, Los Alamitos (2002)
102. Sousa, P., et al.: Enterprise Architecture Modeling with the Unified Modeling Language. In: Enterprise Modeling and Computing with UML. IRM Press (2005)
103. Sowa, J.F., Zachman, J.A.: Extending and formalizing the framework for information systems architecture. IBM Syst. J. 31(3), 590–616 (1992)
104. Srinivasan, K., Jayaraman, S.: Integration of simulation with enterprise models. In: Proceedings of the 29th conference on Winter simulation. IEEE Computer Society, Atlanta (1997)
105. Steen, M.W.A., et al.: Supporting Viewpoint-Oriented Enterprise Architecture. In: Proceedings of the Enterprise Distributed Object Computing Conference, Eighth IEEE International. IEEE Computer Society, Los Alamitos (2004)
106. Stojanovic, Z., Dahanayake, A.: Components and Viewpoints as Integrated Separations of Concerns in System Designing. In: Aspect oriented Design (AOD), Workshop, Delft University of Technology (2002)
107. Tang, A., Han, J., Chen, P.: A Comparative Analysis of Architecture Frameworks. In: Proceedings of the 11th Asia-Pacific Software Engineering Conference. IEEE Computer Society, Los Alamitos (2004)
108. Tarcisius, G., Al-Ekram, R., Ping, Y.: Enterprise Architecture – An Overview (2002)
109. Taylor, K., Palmer, D.: Applying enterprise architectures and technology to the embedded devices domain. In: Proceedings of the Australasian information security workshop conference on ACSW frontiers 2003. Australian Computer Society, Inc., Adelaide (2003)
110. Thornton, S.: Understanding and Communicating with Enterprise Architecture Users. In: Saha, P. (ed.) Handbook Of Enterprise Systems Architecture In Practice, pp. 145–159. Information Science Reference, London (2007)
111. van der Torre, L.W.N., Lankhorst, M.M., ter Doest, H., Campschroer, J.T.P., Arbab, F.: Landscape maps for enterprise architectures. In: Dubois, E., Pohl, K. (eds.) CAiSE 2006. LNCS, vol. 4001, pp. 351–366. Springer, Heidelberg (2006)
112. Tuft, B.: The Changing Role of IT Strategy: Enterprise Architecture Strategies, Meta Group (2001)
113. Vasconcelos, A., et al.: An Information System Architectural Framework for Enterprise Application Integration. In: Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS 2004) - Track 8, vol. 8. IEEE Computer Society, Los Alamitos (2004)
114. Vasconcelos, A., Sousa, P., Tribolet, J.: Information System Architectures (2003)
115. Vasconcelos, A., Sousa, P., Tribolet, J.: Open Issues On Information System Architecture Research Domain: The Vision. In: Proceedings of the 6th International Conference on Enterprise Information Systems (ICEIS 2004) (2004)
116. Wagter, R., et al.: Dynamic Enterprise Architecture. How to make it work. John Wiley & Sons, Inc., Chichester (2005)

117. Wegmann, A.: The Systemic Enterprise Architecture Methodology (SEAM). In: Proceedings of the International Conference on Enterprise Information Systems 2003. Angers, France (2003)
118. Wegmann, A., et al.: Teaching Enterprise Architecture And Service-oriented Architecture in Practice. In: Proceedings of the 2nd Workshop on Trends in Enterprise Architecture Research 2007 (2007)
119. West, D., Bittner, K., Glenn, E.: Ingredients for Building Effective Enterprise Architectures (2002)
120. Whitman, L., Ramachandran, K., Ketkar, V.: A taxonomy of a living model of the enterprise. In: Proceedings of the 33nd conference on Winter simulation. IEEE Computer Society, Arlington (2001)
121. Williams, T.J.: The Purdue Enterprise Reference Architecture (PERA). In: Proceedings of the JSPE/IFIP TC5/WG5.3 Workshop on the Design of Information Infrastructure Systems for Manufacturing. North-Holland Publishing Co., Amsterdam (1993)
122. Wognum, N., Ip-Shing, F.: Maturity of IT-Business Alignment: An Assessment Tool. In: Saha, P. (ed.) Handbook Of Enterprise Systems Architecture In Practice, pp. 221–236. Information Science Reference, London (2007)
123. Zachman, J.A.: A framework for information systems architecture. IBM Systems Journal 26, 276–292 (1987)
124. Zachman, J.A.: Enterprise Architecture: The Issue Of The Century (1996)
125. Zachman, J.A.: Enterprise Architecture: Looking Back and Looking Ahead (1999)
126. Zarvic, N., Wieringa, R.: An Integrated Enterprise Architecture Framework for Business-IT Alignment. In: Proceedings on BUSITAL 2006 - A workshop on Business/IT Alignment and Interoperability, Luxembourg (2006)
127. Langenberg, K., Wegmann, A.: EA: What aspects is current research targeting?, EPFL Technical Report (2004)
128. Esswein, W., Weller, J.: Unternehmensarchitekturen, Grundlagen, Verwendung und Frameworks, Praxis der Wirtschaftsinformatik (2008)
129. ISO 15704, ISO. ISO 15794 - Industrial Automation Systems – Requirements for Enterprise Reference Architectures and Methodologies (2000), http://www.iso.org

Author Index