Services Science
LNCS 7221

George Pallis Mohamed Jmaiel
Anis Charfi Sven Graupner
Yücel Karabulut Sam Guinea
Florian Rosenberg Quan Z. Sheng
Cesare Pautasso Sonia Ben Mokhtar (Eds.)

Service-Oriented Computing – ICSOC 2011 Workshops

ICSOC 2011 International Workshops
WESOA, NFPSLAM-SOC, and Satellite Events
Paphos, Cyprus, December 2011
Revised Selected Papers

Springer

Lecture Notes in Computer Science 7221

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

Services Science

Subline of Lectures Notes in Computer Science

George Pallis Mohamed Jmaiel
Anis Charfi Sven Graupner
Yücel Karabulut Sam Guinea
Florian Rosenberg Quan Z. Sheng
Cesare Pautasso Sonia Ben Mokhtar (Eds.)

Service-Oriented Computing – ICSOC 2011 Workshops

ICSOC 2011 International Workshops
WESOA, NFPSLAM-SOC, and Satellite Events
Paphos, Cyprus, December 5-8, 2011
Revised Selected Papers

Springer

Volume Editors

George Pallis
E-mail: gpallis@cs.ucy.ac.cy

Mohamed Jmaiel
E-mail: mohamed.jmaiel@enis.rnu.tn

Anis Charfi
E-mail: anis.charfi@sap.com

Sven Graupner
E-mail: sven.graupner@hp.com

Yücel Karabulut
E-mail: yuecel.karabulut@sap.com

Sam Guinea
E-mail: guinea@elet.polimi.it

Florian Rosenberg
E-mail: rosenberg@ibm.com

Quan Z. Sheng
E-mail: qsheng@cs.adelaide.edu.au

Cesare Pautasso
E-mail: c.pautasso@ieee.org

Sonia Ben Mokhtar
E-mail: sonia.ben-mokhtar@liris.cnrs.fr

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-31874-0 e-ISBN 978-3-642-31875-7
DOI 10.1007/978-3-642-31875-7
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2012941875

CR Subject Classification (1998): H.3.4, H.3.5, D.2, H.4, C.2, J.1, H.5, K.6

LNCS Sublibrary: SL 2– Programming and Software Engineering

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

This volume contains the proceedings of the scientific satellite events that were held in conjunction with the 2011 International Conference on Service-Oriented Computing, held during December 5–8, 2011 in Paphos, Cyprus. Such satellite events traditionally play a key role in stimulating active exchange and interaction related to the conference topics. This year, the scientific program was particularly rich and addressed various challenging research issues. The selected scientific satellite events were organized around the following three main tracks:

- **Workshop Track**: This ICSOC's workshop program comprised two workshops on current topics, presenting results and work in progress: (1) the Workshop on Engineering Service-Oriented Applications (WESOA 2011), organized for the seventh time in the context of ICSOC, addressed topics from the area of software engineering services; and (2) the 5th Workshop on Non-Functional Properties and SLA Management in Service-Oriented Computing (NFPSLAM-SOC 2011) looked at the management of service level agreements in the context of service-oriented computing.
- **Phd Symposium Track**: The ICSOC 2011 PhD Symposium, as part of the ICSOC conference, was an international forum for PhD students working in the broad areas of service computing, Web services and service engineering to present and discuss emerging research problems and ideas on how to tackle these issues.
- **Demonstration Track**: The ICSOC 2011 Demonstration Program comprised eight practical contributions addressing general or domain-specific challenges of service-oriented computing, and of its transition toward cloud computing.
- **Industry Track**: A key goal of the Industry Track was to expose the academic and industrial research communities to practical real-life problems in the focus areas of ICSOC 2011. Six papers were selected after the review process. These papers describe applications, prototypes or case studies with practical relevance, and report on work that was conducted in collaboration with industry partners. Various topics were covered in the Industry Track such as architecture and modeling of services, service composition, performance analysis as well as crowdsourcing for improving service processes and for knowledge discovery.

A special thanks goes to the satellite event authors, keynote speakers and panelists who together contributed to this important aspect of the conference. It is our great pleasure and privilege to present these proceedings. We hope that these proceedings will serve as a valuable reference for researchers and practitioners working in the service-oriented computing domain and its emerging applications.

February 2012

Mohamed Jmaiel
George Pallis
Anis Charfi
Sven Graupner
Yuecel Karabulut
Sam Guinea
Florian Rosenberg
Michael Q. Sheng
Cesare Pautasso
Sonia Ben Mokhtar

Organization

General Chairs

Mohand-Said Hacid	University of Lyon, France (Honorary General Chair)
George Papadopoulos	University of Cyprus, Cyprus
Winfried Lamersdorf	University of Hamburg, Germany

Program Chairs

Gerti Kappel	Vienna University of Technology, Austria
Hamid Motahari	HP Labs, USA
Zakaria Maamar	Zayed University, UAE

Workshop Chairs

Mohamed Jmaiel	University of Sfax, Tunisia
George Pallis	University of Cyprus, Cyprus

Industry Chairs

Anis Charfi	SAP, Germany
Sven Graupner	HP Labs, USA
Yuecel Karabulut	SAP, USA

Demonstration Chairs

Sam Guinea	Politecnico di Milano, Italy
Florian Rosenberg	IBM Research, USA

Panel Chairs

Youakim Badr	The Pennsylvania State University, USA
Francisco Curbera	IBM T.J. Watson Research Center, USA

PHD Symposium Chairs

Michael Q. Sheng	Adelaide University, Australia
Cesare Pautasso	University of Lugano, Switzerland
Sonia Ben Mokhtar	LIRIS, CNRS, France

Publicity Chairs

Leandro Krug Wives	UFRGS, Brazil
Ivan Bedini	Alcatel-Lucent Bell Labs, Ireland
Yacine Atif	UAE University, UAE
Rainer Unland	University of Duisburg-Essen, Germany

Organizing Committee

Christos Mettouris	University of Cyprus, Cyprus

Publication Chair

Dieter Mayrhofer	Vienna University of Technology, Austria

GECON Workshop Organizing Committee

Jorn Altmann	Seoul National University, South Korea
Omer F. Rana	Cardiff University, UK
Kurt Vanmechelen	Universiteit Antwerpen, Belgium

WESOA Workshop Organizing Committee

George Feuerlicht	UTS, AU and Prague University of Economics, Czech Republic
Winfried Lamersdorf	University of Hamburg, Germany
Guadalupe Ortiz	University of Cadiz, Spain
Christian Zirpins	Karlsruhe Institute of Technology, Germany

NFPSLAM-SOC Workshop Organizing Committee

Flavio De Paoli	Università degli studi di Milano, Italy
Ioan Toma	University of Innsbruck, Austria
Marcel Tilly	European Microsoft Innovation Center, Aachen, Germany
Carlos Pedrinaci	Knowledge Media Institute - The Open University, UK

Table of Contents

Workshop Track

WESOA 2011

NFPSLAM-SOC 2011

Phd Symposium Track

Demonstration Track

Industry Track

Seventh International Workshop on Engineering Service-Oriented Applications (WESOA 2011)

George Feuerlicht[1,2], Howard Foster[3], Winfried Lamersdorf[4], Guadalupe Ortiz[5], and Christian Zirpins[6]

[1] Prague University of Economics
jirif@vse.cz
[2] University of Technology, Sydney
george.feuerlicht@uts.edu.au
[3] City University London
Howard.Foster.1@city.ac.uk
[4] University of Hamburg
lamersdorf@informatik.unihamburg.de
[5] University of Cádiz
guadalupe.ortiz@uca.es
[6] SEEBURGER AG
c.zirpins@seeburger.de

1 Introduction

Rapidly expanding applications of software services, in particular in the context of cloud computing, demand close collaboration of research community and industry practitioners in the development of comprehensive and reliable methodologies and tools that support the entire service systems development lifecycle (SDLC). Development of service-oriented applications presents specific challenges as such applications tend to be process-driven, loosely-coupled, and composed from autonomous services supported by diverse systems. Service-oriented applications typically need to provide multiple, flexible and sometimes situational interaction channels within and beyond organizational structures and processes. Engineering of such software systems requires collaborative and cross-disciplinary development processes, methodologies and tools capable of addressing multiple SDLCs of various service artifacts. There is an urgent need for research community and industry practitioners to agree on comprehensive engineering principles, methodologies and develop tools to support for the entire SDLC of service-oriented applications.

WESOA'11 was the seventh workshop in a series organized in association with the International Conference on Service Oriented Computing (ICSOC). The workshop provides an annual forum for researchers and practitioners in the area of service engineering to exchange ideas and to contribute to evolution of this important field of research. WESOA'11 was held in Paphos, Cyprus on December 5, 2011. We have received sixteen paper submissions for the workshop. Each paper was reviewed by at least three members of the international program committee ensuring high quality of contributions. The workshop started with a keynote presentation by Howard Foster of City University London on Safety, *Certification and Compliance of Software Services*. Thereafter, the technical sessions consisted of eight high-quality papers

G. Pallis et al. (Eds.): ICSOC 2011, LNCS 7221, pp. 1–2, 2012.

representing a rich variety of topics ranging from strategic alignment of business processes to papers discussing service composition, service life-cycle issues, and application of principles and methods of service engineering.

The sessions included presentations by Evan Morrison on *Strategic Alignment of Business Processes*, Cedric Tedeschi on *Decentralized Workflow Coordination through Molecular Composition,* Dominik Meilaender on *Using a Lifecycle Model for Developing and Executing Real-Time Online Applications on Clouds,* Maryam Razavian on *A Pragmatic Approach for Analysis and Design of Service Inventories,* Niels Lohmann*,* on *Artifact-centric modeling using BPMN*, Kristof Hamann on *Migratability of BPMN 2.0 Process Instances,* Casandra Holotescu on *Asynchronous Learning for Service Composition,* and Peter Bartalos on *Engineering Energy-Aware Web Services Toward Dynamically-Green Computing*.

2 Workshop Co-organizers

- Howard Foster, City University London, United Kingdom (Invited PC Co-Chair)
- George Feuerlicht, Prague University of Economics, Czech Republic
- Winfried Lamersdorf, University of Hamburg, Germany
- Guadalupe Ortiz, University of Cádiz, Spain
- Christian Zirpins, SEEBURGER AG, Germany

3 Program Committee

Sudhir Agarwal, KIT Karlsruhe	Mark Little, Redhat
Marco Aiello, University of Groningen	Heiko Ludwig, IBM Research
Sami Bhiri, DERI Galway	Michael Maximilien, IBM Almaden Research
Alena Buchalcevova, Prague University of Economics	Massimo Mecella, Univ. Roma LA SAPIENZA
Javier Cámara Moreno, University of Coimbra	Daniel Moldt, University of Hamburg
Robert Castaneda, CustomWare	Martin Molhanec, Czech Technical University
Anis Charfi, SAP Research CEC Darmstadt	Rebecca Parsons, ThoughtWorks
Jen-Yao Chung, IBM T.J. WatsonResearch	Cesare Pautasso, Universitz of Lugano
Vincenzo D'andrea, University of Trento	Greg Pavlik, Oracle
Florian Daniel, University of Trento,	Pierluigi Plebani, Politecnico di Milano
Keith Duddy, Queensland University of Technology	Franco Raimondi, University College London
Schahram Dustdar, TU Vienna	Wolfgang Reisig, Humboldt-University Berlin
José Luiz Fiadeiro, University of Leicester	Karel Richta, Charles University
Paul Greenfield, CSIRO	Norbert Ritter, University of Hamburg
Birgit Hofreiter, Hochschule Lichtenstein	Nelly Schuster, FZI Forschungszentrum Informatik
Dimka Karastoyanova, University of Stuttgart	Gianluigi Viscusi, Milan Bicocca
Rannia Khalaf, IBM T.J. Watson Research	Olaf Zimmermann, IBM Research Zürich

Acknowledgements. The organizers of the WESOA'11 workshop would like to thank all the authors for their contributions to this workshop, and the members of the program committee whose expert input made this workshop possible. Special thanks go to the invited PC Co-Chair Howard Foster for his effective management of the paper selection process and organization of workshop program. Finally, we thank ICSOC workshop chairs Mohamed Jmaiel and George Pallis for their direction and guidance.

Taming the Cloud: Safety, Certification and Compliance for Software Services

Keynote at the Workshop on Engineering Service-Oriented Applications (WESOA) 2011

Howard Foster and George Spanoudakis

Department of Computing, School of Informatics,
City University London, Northampton Square,
London, England, United Kingdom
{howard.foster.1,g.e.spanoudakis}@city.ac.uk

Abstract. The maturity of IT processes, such as software development, can be and is often certified. Current trends in the IT industry suggest that software systems in the future will be very different from their counterparts today, with an increasing adoption of the Service-Oriented Architecture (SOA) design pattern and the deployment of Software-as-a-Service (SaaS) on Cloud infrastructures. In this talk we discuss some issues surrounding engineering Software Services for Cloud infrastructures and highlight the need for enhanced control, service-level agreement and compliance mechanisms for Software Services. Cloud Infrastructures and Service Mash-ups.

1 Introduction

Aligned with the WESOA workshop theme for this year, this talk discussed the areas of Software Engineering for Cloud, Service-Orientation and Mash-Ups with a focus on safety, security and compliance. Although it is individually challenging to discuss these areas, integrating these together involves further complexity and challenges. The talk specifically aims to highlight the challenges to software engineers, both in engineering discipline at design-time and run-time. We firstly highlighted an interesting case study where architectural decisions in software service design and deployment decisions caused process and resource usage safety issues. Second, we discussed certification for software services and specifically how it enhances a trust mechanism in service discovery and composition. Third, we outlined service policies and constraints, in the form of Service-Level Agreements (SLAs) and how these can be used to monitor services on cloud infrastructures.

2 Safety of Services in Cloud Deployment

Safety for cloud, services and their composition can be viewed from various perspectives. Whilst it is not isolated to service compositions, service design requires some care when deployment is to be made to the cloud and potentially

G. Pallis et al. (Eds.): ICSOC 2011, LNCS 7221, pp. 3–8, 2012.

dynamically changing resource providers and consumers. As an example problem, when enacting a web service orchestration defined using the Web Services Business Process Execution Language (WS-BPEL) we observed various safety property violations. This surprised us considerably as we had previously established that the orchestration was free of such property violations using existing BPEL model checking techniques. In this talk, we described the origins of these violations. They result from a combination of design and deployment decisions, which include the distribution of services across hosts, the choice of synchronisation primitives in the process and the threading configuration of resource containers. We illustrated how model checking can take execution resource constraints into account (see Figure 1). We evaluate the approach by applying it to the above application and are able to demonstrate that a change in allocation of services to hosts is indeed safe, a result that we are able to confirm experimentally in the deployed system. The approach is supported by a tool suite, known as WS-Engineer, providing automated process translation, architecture and model-checking views [2,5,6].

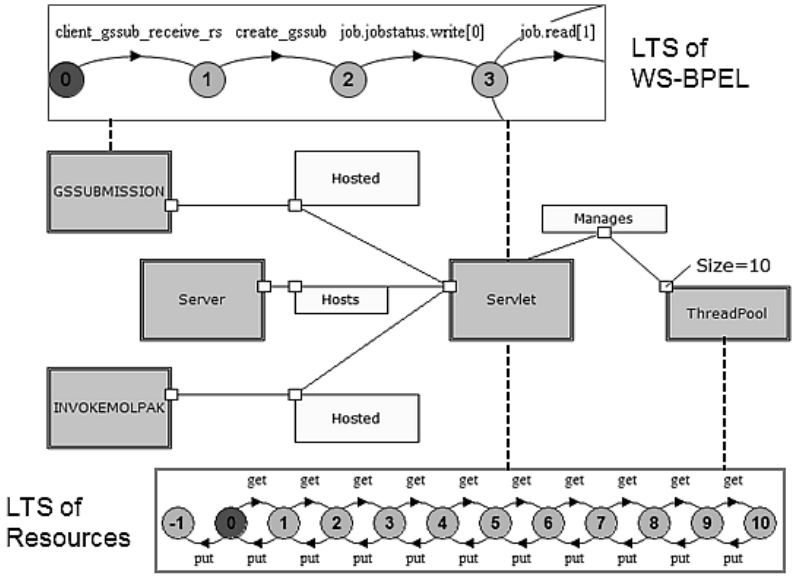


Fig. 1. Example Models for Safety Analysis in Service Deployment

Whilst our focus previously was supporting design-time analysis of service deployment architectures and behavioural models, we also describe how our future work and vision is to provide this at run-time. This involves a collaborative effort between analysis and monitoring. We describe this further in section 5. Our work on analysis however, also raised some interesting points on general service engineering. How do we certify services to be compliant with quality standards?

What are the quality standards that describe this? How are certifications expressed and related between others? What are the safety and security properties we are interested in upholding? These are covered the following parts of the talk.

3 Service Certifications and Cloud Security

Software certification has largely focused on certifying security mechanisms (e.g. ISO/IEC 15408) or product quality metrics (e.g. ISO/IEC 9126*). Our work in certification and assertions for services is part of the EU funded project ASSERT4SOA [8]. ASSERT4SOA aims to provide a general service certification framework, notation and architecture for supporting certificates in service discovery and composition. The ASSERT4SOA language vocabulary is split in to three certification areas: *Evidence-based (ASSERT4SOA-E)*, *Model-based (ASSERT4SOA-M)* and *Ontology-based (ASSERT4SOA-O)*. The three types of certification are used as part of a wider framework to specify and utilise certifications in a service-oriented architecture. As an example usage of this framework, a process for security certificates in service discovery and composition is illustrated in Figure 2.

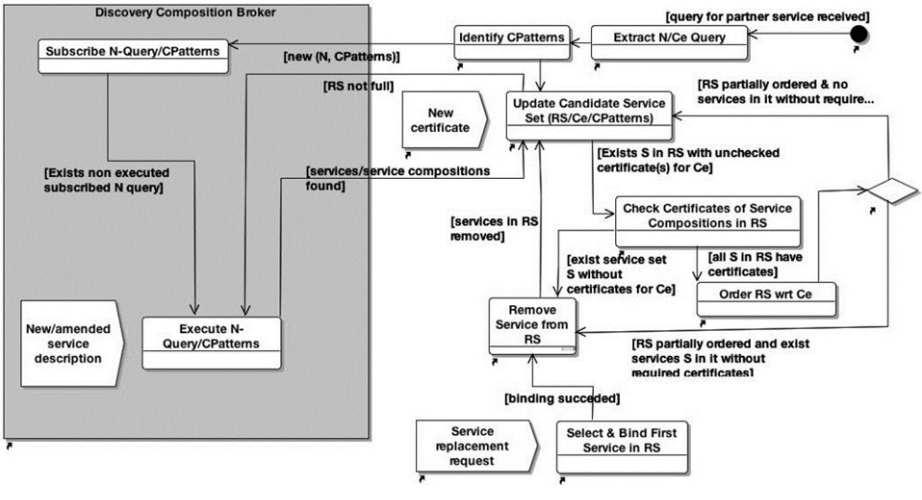


Fig. 2. ASSERT4SOA Framework for Service Composition with Certificates

Note that one particularly challenging area is the trigger from a Service Replacement request. This highlights the dynamic life-cycle of services on "the cloud" and how properties of these services, and the services they rely on, will dynamically change over time and infrastructure changes. Managing these changes and constraining them on Cloud-based *pay-per-usage* model requires an agreement and monitoring architecture.

4 Monitoring Service SLAs in the Cloud

Both Cloud business and infrastructure models emphasise a greater need for accurate levels of service and conformity. With this in mind, specifying and assuring coverage of certain service and infrastructure properties has been undertaken in the EU funded SLA@SOI project [7]. In this project we undertook providing a service monitoring infrastructure with mechanically derived configurations from service-level agreements (SLAs). The architecture for this is illustrated in Figure 3.

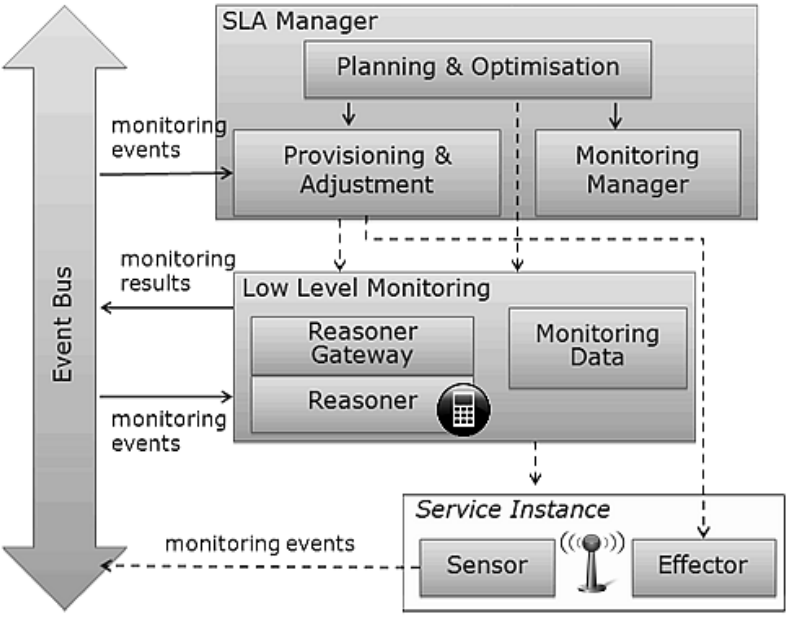


Fig. 3. SLA Monitoring Infrastructure for SOA/Cloud

SLAs are described using the SLA@SOI SLA Model. This includes a vocabulary with terms and constructs for defining complex SLA expressions and constraints. The terms cover a wide range of service and cloud properties, for example, the availability or response time of a service. Both software and infrastructure terms are described, including CPU resource thresholds and virtual machine performance. Certifications provide a way to describe how software meets certain standards. The tests carried out on particular certifications may be drawn from example vocabularies, such as those defined in the SLA@SOI SLA Model. Further details of this work may be found in [3,4].

5 Vision of Service Compliance

Providing both static and dynamic testing tools is a challenge for such techniques discussed previously however we plan to investigate novel yet practical methods for dynamically analysing, monitoring and reacting to violations in such dynamic infrastructures as the Cloud. We discussed mainly a design-time analysis previously, however there is potentially great value to combine service analysis with service monitoring and in the direction of compliance - to ensure certifications are upheld through the software engineering process. Practically, this may be realised through an architecture with checkpoints such as that described by the CBDI [1] and illustrated in Figure 4.

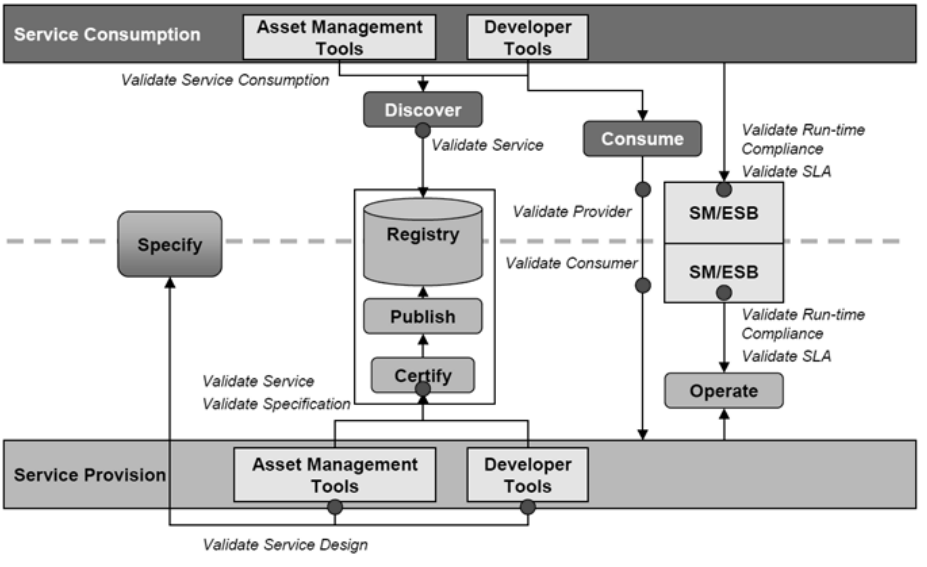


Fig. 4. SOA Architecture with Example Compliance Checkpoints (CBDI 2006)

In summary, we believe that service safety, certification and compliance checking can facilitate a safer cloud of services. As we move towards a more dynamic service-based internet of things, the need for assurance in both providing and consuming services has never been greater. Additionally, new standards of IT compliance may well be delegated to the service engineer to counter such risks as data mis-use and inappropriate service behaviour as a result of ad-hoc service compositions.

Acknowledgements. Our work reported in this keynote has been supported by the EU project ASSERT4SOA - Trustworthy ICT (ICT-2009.1.4).

References

1. Wilkes, L.: Policy Driven Practices for SOA. Presented at the CBDI SOA Seminar (April 2006), http://www.omg.org/news/meetings/workshops
2. Foster, H., Uchitel, S., Magee, J., Kramer, J.: An Integrated Workbench for Model-Based Engineering of Service Compositions. IEEE Transactions on Services Computing (April 2010)
3. Foster, H., Spanoudakis, G.: Dynamic Creation of Service Monitoring Infrastructures. In: Wieder, P., Butler, J.M., Theilmann, W., Yahyapour, R. (eds.) Service Level Agreements for Cloud Computing. Springer, Heidelberg (2011)
4. Foster, H., Spanoudakis, G.: Model-Driven Service Configuration with Formal SLA Decomposition and Selection. In: Proceedings of the 26th ACM Symposium on Applied Computing (SAC), TaiChung, Taiwan (March 2011)
5. Argent-Katwala, A., Clark, A., Foster, H., Gilmore, S., Mayer, P., Tribastone, M.: Safety and Response-Time Analysis of an Automotive Accident Assistance Service. In: Margaria, T., Steffen, B. (eds.) ISoLA 2008. CCIS, vol. 17, pp. 191–205. Springer, Heidelberg (2008)
6. Foster, H., Uchitel, S., Magee, J., Kramer, J.: Model-based Verification of Web Service Compositions. In: IEEE Automated Software Engineering (ASE) 2003, Montreal, Canada (2003)
7. SLA@SOI, A Business-Ready Service-Oriented Infrastructure Empowering The Service Economy in a Flexible and Dependable way, EU Project ICT-2007.1.2, http://sla-at-soi.eu
8. ASSERT4SOA, Advanced Security Service Certificate for SOA, EU Project ICT-2009.1.4, http://assert4soa.eu/

Strategic Alignment of Business Processes

Evan D. Morrison, Aditya K. Ghose, Hoa K. Dam,
Kerry G. Hinge, and Konstantin Hoesch-Klohe

School of Computer Science and Software Engineering,
University of Wollongong
{edm92,aditya,hoa,kgh72,khk789}@uow.edu.au

Abstract. Strategic alignment is a mechanism by which an organization can visualize the relationship between its business processes and strategies. It enables organizational decision makers to collect meaningful insights based on their current processes. Currently it is difficult to show the sustainability of an organization and to determine an optimal set of processes that are required for realizing strategies. Further, there is not a general framework for strategic alignment that can ease this problem. In this article, we propose such a general framework for strategic alignment, which helps develop a clear understanding of the relationships between strategies and business processes. The framework gives organizations an understanding of the relationship between a set of processes and the realization of a set of strategies; it also shows the optimal set of processes that can achieve these strategies.

Keywords: Strategic Alignment, BPM, Strategy Modeling, Requirements Engineering, Governance, Effects, Process Modeling.

1 Introduction

Strategic alignment is a method for understanding the nature of a business through the correlation of business processes and strategies. The use of strategic alignment allows an organization to contemplate its longevity and to find how achievable its visions for the future are. Within the realm of service oriented architectures, verification and validation are significant areas of study. Finding correlations between strategies and business processes are a key component of any SOA methodology [2,4,8,18]. In this article, we build on the foundations of model validation for the description of business process alignment to ensure that there is alignment between processes and strategies. The method of alignment discussed in this article will enable organizations to find if they have the right processes to fulfil their strategies; and thus, will form the basis for understanding sustainable businesses. Our framework for alignment follows from the definition of most specification validation problems [8,9,16] with the extension that we are interested in optimizing the use of processes to fit the given strategies.

During the creation of workflow systems, process designers strive to create process models or designs that can be considered sustainable [2,4,17]. The problem for these activities is in defining the meaning of sustainable process designs

G. Pallis et al. (Eds.): ICSOC 2011, LNCS 7221, pp. 9–21, 2012.

[9,17]. There is a need to describe and to be able to explain why a process model is sustainable and necessary in a given setting [8,9]. By process sustainability, we refer to the long-term effectiveness of a business utilizing efficient processes, measurable through the number of strategies that a business is able to enact. Process models can be viewed as sustainable if they realize part of an organizational strategy. Process models are efficient if when used by an organization they produce optimal results for the organization based on some quality of service (QoS) measure. Organizations are sustainable if all their strategies are realized by a process. The organizational strategy "ensure that employees are happy" and a process designed to make employees happy can be used to illustrate this point. The process would be aligned to the strategy as it realizes the strategy and hence should be considered sustainable. If there are two such processes for making employee's happy, then the optimal process is the process that satisfies a desired QoS description, such as, make employee's happy *quickly*. By identifying the points of interaction between processes and strategies analysts are able to tell if the processes that they have designed are sustainable.

Results from this work hold numerous benefits for designers who ask *What?* and *How?* questions, such as *What strategy does this process seek to satisfy?* and *How is this strategy realized?* Through the use of the alignment framework presented in this article, analysts will be able to describe and explain a specific process model's sustainability. The framework that we propose also provides a mechanism to compute the most optimal model of alignment, which shows the best way to realize given strategies in an organization.

The contributions of this article are as follows. First, we propose a framework that grants business analysts the ability to correlate processes with strategies. Secondly, we describe how an organization can find how many of its strategies are realizable by its current processes. Finally, we show how to compute the most optimal set of processes within the organizational process portfolio to satisfy the organizational strategies.

These contributions are discussed in the article in the following order. In §2, we provide a background of the tools and languages that form the basis of this alignment framework. In §3, we provide an example scenario that describes a generic human resources department. Then in §4 we present the strategic business process alignment framework. Through this work, we have been developing a toolkit that provides automated support of many of the concepts in this framework, which we present in §5. We compare our work to existing literature in §6, then conclude and position our future work in §7.

2 Background

In this section, we will introduce the set of languages used to describe process models and strategies. In the follow sections, we will use these languages to form a crisp description of strategic business process alignment.

2.1 Semantic Process Effects

A business process model represented in the Business Process Modeling Notation (BPMN)[1] is a collection of activities, gateways, events, sequence flows, pools, swim lanes, and message flows. Semantic effect annotations [7] offer a means to reason over business process models. By reasoning with process effects, we are able to capture the organizational operation model, i.e., "what does this process do?" This is important as it allows us to understand what happens as a result of a business process execution; and what execution scenarios a process designer has created for the organization. In other approaches that rely on syntactical process analysis, no information as to what processes do can be extracted from the process models. This makes pure syntactic analysis difficult when attempting to answer *"what"* questions about process models.

Previous work in this area [7] has described a method for semantic annotation of business processes. This is an effective way of adding semantic descriptions to process models as it produces reusable artefacts that can be reasoned over. To construct semantically annotated business process models, analysts annotate activities in the model with descriptions of the changes that occur as a result of the activities execution. Such results are referred to as immediate effects of an activity. For example (see Fig.1), an activity *Check employee database for suitable replacement* within a human resources process model could have the immediate effect: *ConfirmedEligibility*. Similarly, the event *no suitable replacement found* has an immediate effect: $\neg$*HolidayProvisioned*. See figure.1 for the process model of this example. We represent each effect as a proposition and consider a set of effects as a sentence constructed by the conjunction of the propositions in the set. We denote a singleton immediate effect with one effect on its own (e.g. α) and an immediate effect with multiple effects (e.g. $\alpha \wedge \beta$) as a set of effects (e.g. $\{\alpha, \beta\}$).

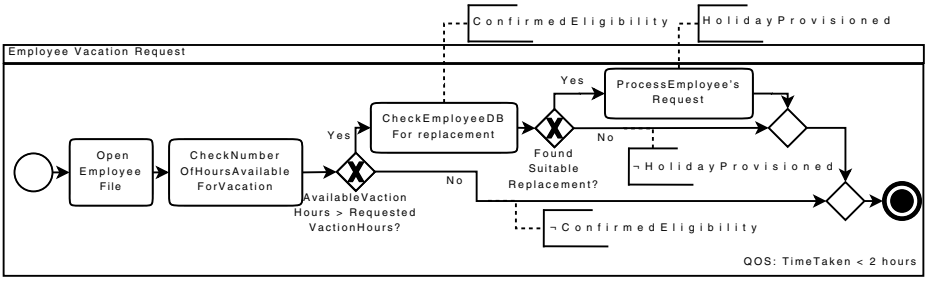


Fig. 1. Employee Vaction Request Process

Each annotation can then be accumulated using a function to produce a semantic description of the process model. Let e_a be a set of effects (or a singleton immediate effect) associated with an activity a within a process P. Given two

[1] See http://www.bpmn.org for full specifications.

sets of effects e_i and e_j, let a function $acc(e_i, e_j)$ (defined in [7]) return the accumulation of both immediate effects which is a set of possible effect scenarios.

An *effect scenario* ϵ is the result of accumulation from a start event ψ in a process to the current activity a, where for each pair of sequential activities, the immediate effect of the first activity in sequence is combined with the immediate effect of the next activity in sequence resulting in a cumulative effect for the pair. The cumulative effect of the pair is then accumulated with the immediate effect of the next activity in sequence, and so on until all activities in the sequence have been accumulated over. For each activity there may be multiple effect scenarios, as these show possible execution instances through the process model.

An *end effect scenario* is an effect scenario associated with an *end event* in a process, i.e. if ϕ is an end event then an effect scenario ϵ associated with ϕ is an end effect scenario. The set of end effect scenarios in a process model is denoted E_P.

A *pathway* in a process model P from the start event ψ to an activity, event or gateway a is a sequence of activities, events or gateways that can be executed in an unbroken sequence in a manner conformant to the model P. We will refer to points in the model $a_i \in a_1, \ldots, a_n$ occuring some point in the model a_j *before* or *after* other model items in a path.

In this article, we will consider strategic alignment using a notion of process composition. This concept is required for describing business process alignment, as normally, we have found that business process models typically do not realize strategies by themselves, because strategies are described in more general language than process models. For example, a business process that describes a set of activities for evacuating a building will not necessarily satisfy an organizational goal to ensure that employees are safe. In our framework we leverage a composition of processes that contribute to the safety strategy of the organization. There is a general need within businesses to connect similar processes and services that meet the needs and demands of different functional requirements [5]. Processes can be composed using either parallel or sequential process semantics, where the parallel joins have corresponding semantics to a BPMN *AND* gateway. A sequential composition has similar semantics to sequential activities within BPMN joined by sequence links.

When discussing process models, we refer to a *process portfolio* [14] as an organization-wide collection of business process models. Each process in a process portfolio describes the capabilities and activities involved in the execution of each process model. Given our description of process effect accumulation, we will be considering alignment between single processes and strategies, as well as alignment between composed processes and strategies. Given a process portfolio $\mathcal{P}$, we shall use the term *composite process portfolio*, denoted by $\mathcal{C}_\mathcal{P}$, to describe the set of all possible compositions of processes in $\mathcal{P}$.

2.2 Strategy Modeling Language

In our previous work [6], we have proposed a language that can be used by senior executives for describing organizational strategies. This language is called

the strategy modeling language[2] (SML). The core modeling elements of SML are: Functional Goals, Plans, and Optimization Objectives. Goals are general desired outcomes that organizations want to meet and when described in SML, can be evaluted to be in the boolean state of either fulfilled or not fulfilled. For example, the goal *Encourage the use of an employee's holiday period*, the goal can be evaluated if there is a process that has an effect that results in *HolidayProvisioned*. Each plan in SML, describes milestones in an organizational strategy. Where the achievement of goals in sequence are key steps that must be completed in a particular order. Plans may follow tactical decisions that describe a plan of progression that will achieve certain higher level goals. For example, a plan *"Ensure that staff are the best in the industry may be shown as a sequence of goals"*: **First** *maintain_high_employee_morale*, **then**, *maintain_ongoing_training*. An optimization objective in SML is used to discriminate preferences for strategic outcomes. Based on a goal to encourage holiday usage, an optimization object may be *minimize wait time for holiday approval; min(WT)*. In this next section, we will give an example of strategies that can be expressed in SML.

3 Motivating Example

Consider a human resources department. There are three components to this example. First, there is the the strategic landscape (i.e., the strategies the dept. seeks to realize), then a HR Knowledge base and finally a set of business processes.

Strategic Landscape. The strategies of the department, described in SM, are as follows:

- (Goal) Encourage the use of an employee's holiday period *HolidayProvisioned*
  - (Optimization Objective) Minimize wait time for holiday approval min(WT) (where WT is wait time).
- (Goal) Maintain retention of high-quality staff *EmployeeRention* $\wedge$ *HighlySkilledWorker*
- (Plan) Ensure that staff are the best in the industry `First` *maintain_high_employee_morale* `then`, *maintain_ongoing_training*
  1. (Goal) Maintain high employee morale *maintain_happy_employee*
  2. (Goal) Maintain ongoing training *maintain_training_provided*

HR Knowledge Base. A domain specific knowledge base that describes the HR department. Knowledge base rules are written as a logical consequence (read $A \Rightarrow B$ as material implication).

- *HappyEmployee* $\wedge$ *SalaryPaid* $\Rightarrow$ *EmployeeRention*

[2] The strategy modeling language (SML) has been implemented in a tool and has been used to construct strategy models for both banking organizations and government agencies - see `http://www.dlab.uow.edu.au/crc` for details of project.

- $\neg SalaryPaid \Rightarrow \neg HappyEmployee$
- $ConfirmedEligibility \wedge \neg HolidayProvisioned \Rightarrow \neg HappyEmployee$
- $ConfirmedEligibility \wedge HolidayProvisioned \Rightarrow HappyEmployee$
- $TrainingProvided \Rightarrow HighlySkilledWorker$

Business Process Models. Fig.1 and Fig.2 (in the appendix) describe the processes in BPMN with fragments of semantic effect annotation (complete annotation of these processes would be too large to describe in this paper). This example illustrates how even a basic set of effect annotation fragments can deliver considerably desired values. The four processes of interest include an automated vacation request process, a manual vacation request process, a salary payment process and a training process.

4 Strategic Alignment of Business Processes

The realization relationship between business process models and goals is critical to strategic alignment analysis and will be defined in this section. We will then expand realization into a relationship that shows alignment between strategies and processes.

Definition 1 (Realization)
A process P with a set of end effect scenarios E_P, realizes a goal G, if and only if an end effect scenario of P entails G, i.e., $\exists \epsilon \in E_P$ s.t. $\epsilon \models G$. We will write: P alignedTo *G if this is the case.*

Consider a set of process models $\{P_1, \ldots, P_n\}$ in a process portfolio $\mathcal{P}$. Given a goal G, we want to determine if the process portfolio $\mathcal{P}$ is aligned to the goal G. Trivially, we have the following basic test: if $\exists P \in \mathcal{P}$ s.t. P *alignedTo* G then, $\mathcal{P}$ is aligned to G; however, we also need to consider the possible compositions of the processes in $\mathcal{P}$. If a goal can be realized by a process in the composite process portfolio then the process portfolio is aligned to the goal.

Definition 2 (Alignment with Goals)
Let $\mathcal{P}$ be a process portfolio, let $\mathcal{C}_P$ be the composite process portfolio derived from $\mathcal{P}$ and let $\mathcal{G}$ be a set of goals. $\mathcal{P}$ is aligned to a single goal G iff $\exists P \in \mathcal{C}_P$ s.t. P alignedTo *G. This is denoted $\mathcal{P}$* alignedTo *G. We will say $\mathcal{P}$* alignedTo *$\mathcal{G}$ iff $\forall G \in \mathcal{G}$. $\mathcal{P}$* alignedTo *G.*

Strategic plans are sequences of strategic goals (or other plans). Each plan describes milestones in an organizational strategy model. Where each goal in the sequence must be achieved before its successor goal. A plan in a strategy model is a temporal sequence of goals.

Definition 3 (Alignment with Plans)
Let a plan L be a sequence of goals $\langle G_1, \ldots, G_n \rangle$. For the plan to be completely realized by a process model (or process models) each pair of consecutive goals $\langle G_i, G_j \rangle$ in the plan must be realized. A plan is realized and aligned to a set of processes if all consecutive goal pairs in the plan are realized. Pairs of goals are realizable in the following ways:

1. *Given two processes P_k and P_l, where the processes can be composed in the sequence $\langle P_k, P_l \rangle$ to form process P_m, if P_k realizes G_i (but not G_j) and P_m realizes $G_i \wedge G_j$ then the process composition P_m realizes the goal pair.*
2. *Given a semantically annotated process model P_n, where there is an activity a with effect scenario ϵ_a that entails the goal G_i and there is an activity b with effect scenario ϵ_b, that occurs in the pathway after activity a, that entails $G_i \wedge G_j$ and there is an end effect scenario of process P_n that entails $G_i \wedge G_j$ then the process P_n realizes the goal pair. The effect scenario ϵ_a must not entail $G_i \wedge G_j$, otherwise the realization order of the goals will be incorrect.*

If the above criteria are met and each goal in L is realized in sequence then the process P *realizes* L. This plan based realization can be incorporated in the alignment model shown in Definition 2.

To compute optimization objective Alignment, given a strategy G, and two processes P and P' with alignment relationships P *alignedTo* G and P' *alignedTo* G. We need to add a mechanism for determining which process is a better fit for the strategy. To do this, we refer to a process capability function that computes the value for processes satisfying a particular strategy; similar functions and capabilities are shown in [11]. The function will return the best process from a collection of processes that can satisfy the strategy with a given objective function. For example, consider an organizational optimization objective O : *'minimize cycle time'* applied to a functional goal encouraging the use of vacation time. Process P may be a manual process that requires the employee to submit leave request forms and find their own replacements, and process P' may be an automated process that automatically selects replacement employees and stream lines the approval process. A QoS execution description for process P may be *Time < 2 days*, and the QoS execution description for process P' may be *Time < 2 hours*. Provided that there are no alternative QoS objectives, then the selection function will select process P' as being the optimal process to satisfy the goal.

Definition 4 (Alignment with Optimization Objectives)
Given a strategy G, an optimization objective O, and two realization scenarios P alignedTo *G and P'* alignedTo *G, then we refer to the optimal candidate process for realization as the most optimally aligned process P alignedOptimallyTo G which is the process that is more preferred based on the optimization objective, i.e. $P \leq_O P'$ iff P, satisfies the optimization objective O better than P'. Similarly, if for a strategy G, there is a set of processes in a process portfolio $\mathcal{P}$ that optimally realizes the strategy, then the realization is denoted $\mathcal{P}$ alignedOptimallyTo G.*

We observe that this selection of optimal processes has been discussed in other research such as in [15]. Using the previous definitions of goal alignment, plan alignment and optimization objective alignment, we can now tie together an alignment definition that can be used to describe strategic business process alignment.

Definition 5 (Strategic Alignment)
Let $\mathcal{C}_\mathcal{P}$ be the set of the composed of processes of $\mathcal{P}$. Let $\mathcal{G}$ be a collection of strategies. $\mathcal{P}$ alignedOptimallyTo $\mathcal{G}$ iff:

1. *For each* $G \in \mathcal{G}$*: (completeness)*
   (a) $\exists \mathcal{P}' \subseteq \mathcal{C}_{\mathcal{P}}.\ \mathcal{P}'\ alignedOptimallyTo\ G$
   (b) *There is no* $\mathcal{P}'' \subset \mathcal{P}'$ *where* $\mathcal{P}''$ *satisfies condition a. (realization minimality)*
   (c) $\neg\exists P \in \mathcal{C}_{\mathcal{P}}.(P \wedge G \models \bot)$ *(consistency)*
2. *There is no* $\mathcal{P}^* \subset \mathcal{P}$ *where* $\mathcal{C}_{\mathcal{P}^*}$ *satisfies condition 1. (alignment minimality)*

It should be noticable that there are differences in purpose between Definition 5 and Definition 2, as minimality conditions are missing from Definition 2. In this setting we argue that finding the best set of processes that are able to meet an entire organizational strategy is of great importance for both executives and analysts.

We will now step through an example of alignment between an organizations business processes and strategies.

Example 1 (Strategic Alignment Example)
Recall from the motivating example §3 there are a number of strategic goals to be realized. In the process portfolio, a number of processes are available for analysis to test if they can be utilized in optimal realization of the organizational strategies. For each process there is a QoS metric for `TimeTaken` annotated at the bottom right of each model.

We must ensure that all strategic goals are realized by the processes in our process portfolio.

First consider the goal: *Encourage the use of an employee's holiday period* $\rightarrow$ *HolidayProvisioned* with the optimization objective *minimize wait time for holiday approval*, to which the *Employee Vacation Request* is aligned as the effects of this process realize the goal condition and the QoS variable for *Employee Vacation Request* is minimal compared to the alternative goal realizing process *Manual Employee Vacation Request*.

Next, we consider the goal: *Maintain retention of high-quality staff* $\rightarrow$ *EmployeeRention* $\wedge$ *HighlySkilledWorker*. From the knowledge base *EmployeeRention* is achieved if there are processes that ensure *HappyEmployee* $\wedge$ *SalaryPaid*.

The *Salary Payment Process* has the effect *SalaryPaid* and the *Employee Vacation Request* process ensures that the effect *HappyEmployee* is fulfilled. *Training Process* has the effect *HighlySkilledWorker* which completes the requirements for the goal to be realized.

Finally, we have a plan with two subgoals: *Ensure that staff are the best in the industry*. To realize this plan, we construct a composed process where we attempt to satisfy each goal.

For the sub-goal:*Maintain ongoing training* $\rightarrow$ *TrainingProvided* the *Training Process* is aligned as the effects of this process realize the goal condition.

The sub-goal: *Maintain high employee morale* $\rightarrow$ *HappyEmployee* can be aligned with *Employee Vacation Request* as the effects of this process realize the goal condition.

The process composed of the Training Process and the Employee Vacation request realizes the plan.

On review of this example, we can determine there is no smaller set of processes we could use to realize all the organizational goals from the example case. As a final analysis on this alignment, we can suggest to the organization that it could drop the manual employee vacation request process.

5 Implementation

To demonstrate the use of our framework we have sought to extend the functionality of Process Seer through a text based toolkit without BPMN modeler support. Currently the tool[3] (shown in Figure 3 in the appendix) is able to load and test process models for consistency against a rule base. The tool builds sequential and parallel process compositions, then process seer [7] style effect accumulation can be computed on the composed process models to find composition end effect scenarios.

6 Related Work

Koliadis et. al. [9] have proposed a framework for aligning business processes to services capabilities. The framework uses semantic effect accumulations over BPMN models to describe relationships mapping effect scenarios to service outcomes. Our framework differs from the framework for alignment in [9] not only through much more detailed and extensible formal descriptions, but also in that we use the strategy modeling language as a basis for goal relations and we also consider ranked realization. The precursor to [9] is described in [8] where Koliadis and Ghose introduce the notion of relating goals (functional goals - from an and/or decomposition tree) with the accumulated effects of processes. This article describes the fundamental relationship between goals and effects showing how processes are related to requirements. Secondly, the article introduces satisfaction goals by the semantic effects of a process. Satisfaction is based on the relationship between process trajectories (or scenario pathways).

Zirpins et. al. [19] have described the alignment of processes to services using capabilities and role based relationships. The work in [19] provides an excellent service adaptation environment that could be leveraged with this work and work in [17] to describe a capability based change management framework.

In the wider spectrum of methods for relating strategic level goals to business processes, Anton [1] has described an approach to process alignment to e^3 value models through a series of model transformations. The primary focus of Anton's work is on goals and the analysis of what role they play within an organization. In [1], strategy or high level goals are refined to operational goals and are then typed using general goal subdivisions like maintain, achieve, etc. The article introduces a basic notion of constraint satisfaction and process activity ordering for goal plan realization. This work is still in its early phases and does not distinguish between process activities, activity goals, and goal refinements of strategic goals.

[3] For source and further implementation details see http://www.dlab.uow.edu.au/textseer/

In [3], Cardoso et. al. have shown a method for eliciting non-functional goals from business processes (with a practical case study in a Brazilian hospital). The authors provide a method for the construction of goal decomposition trees, and then provide a method for composing multiple trees to describe organizational strategy.

For both [1,3] the work appears to be lacking descriptive details beyond a methodology for constructing candidate models of business/strategy relationships. Neither framework has a method for assessing the correctness of models constructed with their implementations.

A framework for goal operationalism has been rigorously constructed by Leiter, Ponsard et. al. [10,13] showing a crisp goal satisfaction framework that can be used to describe the satisfaction of software systems over time.

Pijpers et. al. have presented a framework and methodology for modeling business strategies called the e^3 force in [12]. The e^3 force examines three perspectives of an organization, one of which is focused on business strategy modeling. The other forces in place on an organization are also modeled (these include the value creation perspective and the IT architecture perspective). The use of separation of concerns in Pijper's work aids in clarifying discussions between relevant stakeholders.

Through the literature reviewed, it has become abundantly clear that there needs to be a link between business process models and strategies. The work that is presented in this article provides the next logical and innovative development towards a formalization of the relationships that should exist in any general SOA framework.

7 Conclusion

In this article, we have described alignment as a realization relation between a set of process models and a set of strategies. We have presented a formal framework that can be used to show optimal strategic alignment within an organizational context. The framework contains a set of methods for correlating process models to functional goals, strategic plans and optimization objectives. Further, the result of using strategic alignment to determine the alignment between strategies and processes shows an organization the optimal set of processes from a process portfolio that would help them realize their strategies. The results of this work will provide organizational decision makers with a device to understand sustainability in an operational context. The framework that we have presented contributes to a better understanding of strategic business process alignment and further tool support will equipped decision makers with a device to understand sustainability in an operational context. In future work, we will look to extending the definitions of strategic alignment and then show a method for discovering business processes that can meet organizational strategies, attempting to provide a method for rapid business infrastructure development.

Acknowledgements. Work on this project has been supported by the Australian CRC for Smart Services.
http://www.smartservicescrc.com.au/, paper tracking number A10155.

References

1. Anton, A.: Goal-based requirements analysis. In: International Conference on Requirements Engineering, pp. 136–144 (1996)
2. Bleistein, S.J., Cox, K., Verner, J.: Validating strategic alignment of organizational it requirements using goal modeling and problem diagrams. Journal of Systems and Software 79(3), 362–378 (2006)
3. Cardoso, E.C.S., Almeida, J.P.A., Guizzardi, G., Guizzardi, R.S.S.: Eliciting Goals for Business Process Models with Non-Functional Requirements Catalogues. In: Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Soffer, P., Ukor, R. (eds.) BPMDS 2009 and EMMSAD 2009. LNBIP, vol. 29, pp. 33–45. Springer, Heidelberg (2009)
4. Edirisuriya, A., Johannesson, P.: On the Alignment of Business Models and Process Models. In: Ardagna, D., Mecella, M., Yang, J. (eds.) BPM 2008 Workshops. LNBIP, vol. 17, pp. 68–79. Springer, Heidelberg (2009)
5. Feuerlicht, G.: Simple Metric for Assessing Quality of Service Design. In: Maximilien, E.M., Rossi, G., Yuan, S.-T., Ludwig, H., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6568, pp. 133–143. Springer, Heidelberg (2011)
6. Ghose, A.K., Lê, L.S., Hoesch-Klohe, K., Morrison, E.: The Business Service Representation Language: A Preliminary Report. In: Cezon, M., Wolfsthal, Y. (eds.) ServiceWave 2010 Workshops. LNCS, vol. 6569, pp. 145–152. Springer, Heidelberg (2011)
7. Hinge, K., Ghose, A.K., Koliadis, G.: Process seer: A tool for semantic effect annotation of business process models. In: 13th IEEE International EDOC Conference (2009)
8. Koliadis, G., Ghose, A.: Relating Business Process Models to Goal-Oriented Requirements Models in KAOS. In: Hoffmann, A., Kang, B.-h., Richards, D., Tsumoto, S. (eds.) PKAW 2006. LNCS (LNAI), vol. 4303, pp. 25–39. Springer, Heidelberg (2006)
9. Koliadis, G., Ghose, A., Padmanabhuni, S.: Towards an enterprise business process architecture standard. In: IEEE Congress on Services, pp. 239–246 (2008)
10. Letier, E., van Lamsweerde, A.: Deriving operational software specifications from system goals. SIGSOFT Softw. Eng. Notes 27(6), 119–128 (2002)
11. Ortiz, G., Bordbar, B.: Aspect-oriented quality of service for web services: A model-driven approach. In: ICWS, pp. 559–566 (2009)
12. Pijpers, V., Gordijn, J., Akkermans, H.: Business strategy-it alignment in a multi-actor setting: a mobile e-service case. In: Proceedings of the 10th International Conference on Electronic Commerce (2008)
13. Ponsard, C., Massonet, P., Molderez, J., Rifaut, A., Lamsweerde, A., Van, H.: Early verification and validation of mission critical systems. Formal Methods in System Design 30, 233–247 (2007)
14. Rosemann, M.: The service portfolio of a bpm center of excellence. In: Handbook on Business Process Management 2, pp. 267–284 (2010)
15. Vilenica, A., Hamann, K., Lamersdorf, W., Sudeikat, J., Renz, W.: An Extensible Framework for Dynamic Market-Based Service Selection and Business Process Execution. In: Felber, P., Rouvoy, R. (eds.) DAIS 2011. LNCS, vol. 6723, pp. 150–164. Springer, Heidelberg (2011)
16. Wang, H.-L., Ghose, A.K.: On the foundations of strategic alignment. In: Proc. of the 2006 Australia and New Zealand Academy of Management Conference (2006)

17. Wang, H.L., Ghose, A.K.: Green strategic alignment: Aligning business strategies with sustainability objectives. In: Unhelkar, B. (ed.) Handbook of Research in Green ICT, pp. 29–41 (2010)
18. Zairi, M.: Business process management: a boundaryless approach to modern competitiveness. Business Process Management Journal 3(1), 64–80 (1997)
19. Zirpins, C., Piccinelli, G.: Evolution of service processes by rule based transformation. In: IFIP International Conference on Digital Communities in a Networked Society, pp. 287–305 (2004)

A Appendix

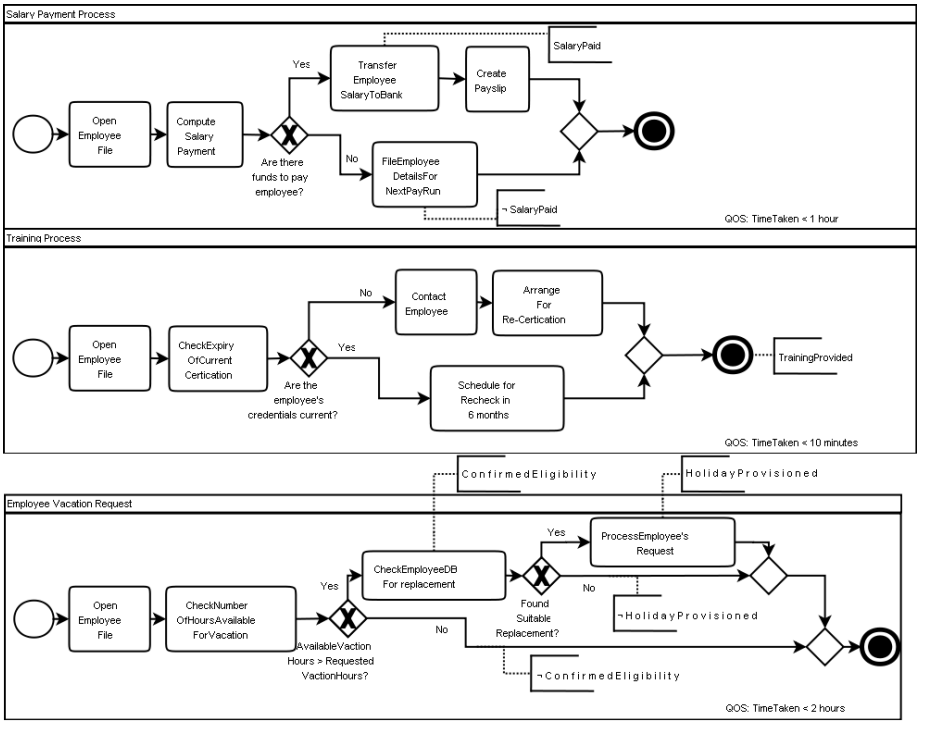


Fig. 2. Process Portfolio Examples : Manual Employee Vaction Request Process, Training Process, Salary Payment Process

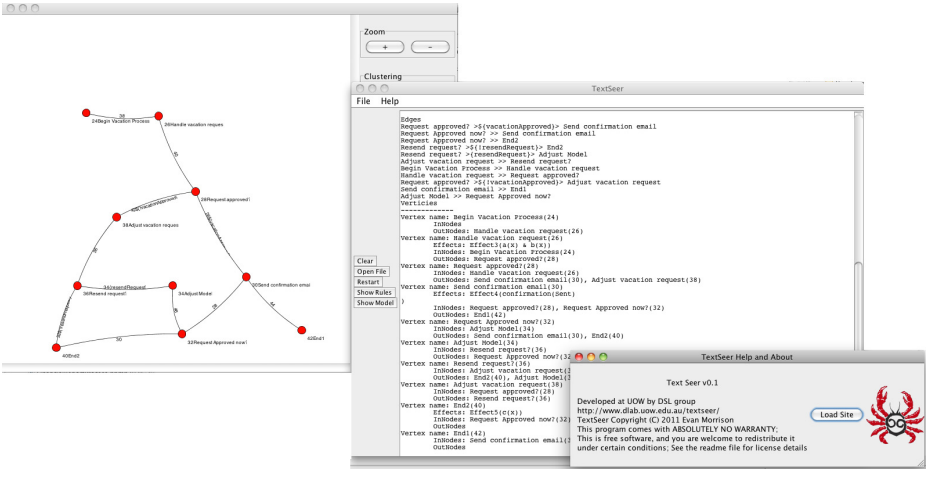


Fig. 3. Screenshot of Toolkit[a]

[a] For source and further implementation details see `http://www.dlab.uow.edu.au/textseer/`

Decentralized Workflow Coordination through Molecular Composition

Héctor Fernández[1], Cédric Tedeschi[2], and Thierry Priol[1]

[1] INRIA, France
[2] IRISA, University of Rennes 1 / INRIA, France
{hector.fernandez,cedric.tedeschi,thierry.priol}@inria.fr

Abstract. The dynamic composition of loosely-coupled, distributed and autonomous services is one of the new challenges of large scale computing. Hence, service composition systems are now a key feature of service-oriented architectures. However, such systems and associate languages strongly rely on centralized abstractions and runtime, what appears inadequate in the context of emerging platforms, like (federation) of clouds that can shrink or enlarge dynamically. It appears crucial to promote service composition systems with a proper support for autonomous, decentralized coordination of services over dynamic large-scale platforms. In this paper, we present an approach for the autonomous coordination of services involved in the execution of a workflow of services, relying on the analogy of molecular composition. In this scope, we trust in the chemical programming model, where programs are seen as molecules floating and interacting freely in a chemical solution. We build a library of molecules (data and reactions) written with HOCL, a higher-order chemical language, which, by composition, will allow a wide variety of workflow patterns to be executed. A proof of concept is given through the experimental results of the deployment of a software prototype implementing these concepts, showing their viability.

1 Introduction

Today, Information Technology (IT) applications tend more and more to be designed as a composition of services, following the wide adoption of service-oriented architecture (SOAs). These services are composed to form more complex services, commonly referred to as *workflows*. The problem of service composition gained recently a lot of attention from the industrial and academic research communities, both trying to define adequate service-based software engineering methods. Concomitantly, cloud computing has emerged as the next generation computing platform. One key property of clouds is *elasticity*, *i.e.*, the possibility for cloud users to dynamically extend the number of resources at their disposal.

Thus, future service composition systems should provide ways to express both workflows and platform characteristics. In particular, those systems must be able to deal with the high degree of parallelism and distribution of services over elastic platforms (especially in the case of federations of clouds). However,

G. Pallis et al. (Eds.): ICSOC 2011, LNCS 7221, pp. 22–32, 2012.

current service composition systems highly rely on a centralized vision of the coordination, failing to provide the right abstractions to express decentralization, and leading to various weaknesses at runtime such as poor scalability, availability, and reliability [3].

Lately, nature metaphors, and more specifically artificial chemistries [8], have been identified as a promising source of inspiration to express autonomous service coordination [20]. Among them, the *chemical programming paradigm* is a high-level execution model, where a computation is basically seen as a concurrent set of reactions consuming some molecules of data interacting freely within a *chemical solution* to produce new ones (resulting data). Reactions take place in an implicitly parallel, autonomous and decentralized manner. Recently, the Higher-Order Chemical Language (HOCL) [4] raised the chemical model to the higher-order, providing a highly-expressive paradigm: rules can apply on other rules, programs dynamically modifying programs, opening doors to dynamic adaptation and autonomous coordination [5].

Contribution. In this paper, we leverage the chemical analogy and extend it to *molecular composition* to model the decentralized execution of a wide variety of workflow structures, *a.k.a*, *workflow patterns* [19]. In our approach, services, as well as their control and data dependencies, are molecules interacting in the workflow's chemical solution. Chemical rules – higher-order molecules – are in charge of its decentralized execution, by triggering the required reactions locally and independently from each others. These local reactions, together, realize the execution of the specified workflow. A software prototype implementing these abstractions, has been developed, based on the HOCL chemical language. Experimentations over an actual platform connecting several geographically distributed clusters, have been conducted. For the sake of comparison, the prototype was developed in two versions – centralized and decentralized – and the performance gain obtained with the decentralized version over the centralized version are discussed in details.

Outline. Section 2 presents the background of our work: the chemical programming paradigm and its distributed architectural framework. Section 3 details our decentralized coordination model and language. Section 4 presents the software prototype, the experimental campaign and its results. Section 5 presents related works and Section 6 concludes the paper.

2 Chemical Computing

Bio-chemical analogies have been shown recently to be highly relevant in the design of autonomic service architectures [20]. In particular, the chemical programming paradigm exhibits the properties required in emerging cloud-based service architectures.

2.1 Chemical Programming Paradigm

According to the chemical metaphor, molecules (data) float in a chemical solution, and react according to reaction rules (program) producing new molecules

(resulting data). At runtime, reactions take place in an implicitly parallel and autonomous way, according to local conditions, without any central coordination, artificial ordering or serialization, until no more reactions are possible, a state referred to as *inertia*. This programming style allows to write programs keeping only the functional aspects of the problem to be solved.

In artificial chemistries [8], the solution is usually formally represented by a multiset of molecules, and reactions between molecules are rules rewriting this multiset. Recently, higher-order chemical programming has been proposed, through HOCL (*Higher-Order Chemical Language*) [4]. In HOCL, every entity is a molecule, including reaction rules. An HOCL program is basically a multiset of atoms, where atoms can be constants, tuples, sub-solutions, or reaction rules. A reaction involves a reaction rule **replace** P **by** M **if** V and a set of molecules denoted N satisfying the pattern P and the reaction condition V. The reaction consumes N, to produce a new molecule M. The alternative **replace-one** P **by** M **if** V can react only once, and is deleted in the reaction. Rules can be named. Let us consider the simple HOCL program below that extracts the maximum even number from a set of integers.

$$
\begin{array}{l}
\textbf{let } selectEvens = \ \textbf{replace } x, \omega \ \textbf{ by } \omega \textbf{ if } x\%2 \ != 0 \\
\textbf{let } getMax = \ \textbf{replace } x, y \ \textbf{ by } x \textbf{ if } x \geq y \\
\textbf{in } \langle \\
\quad \langle selectEvens, 2, 3, 5, 6, 8, 9 \rangle, \\
\quad \textbf{replace-one } \langle selectEvens = s, \omega \rangle \textbf{ by } getMax, \ \omega \\
\rangle
\end{array}
$$

The *selectEvens* rule removes odd numbers from the solution, by its repeated reactions with an integer x, ω denoting the whole solution in which *selectEvens* floats, deprived of x. The $getMax$ rule reacts with two integers x and y such that $x \geq y$ and replaces them by x. When triggered repeatedly, this rule extracts the maximum value from a solution of integers. The solution is composed by (i) a sub-solution containing the input integers along with the *selectEvens* rule, and (ii) a higher-order rule (third line of the solution) that *opens* the sub-solution, extracts the remaining (even) numbers and introduces the $getMax$ rule.

This program leverages the higher-order: a sub-solution can react with other molecules as soon as it has reached inertia itself. In other words, the $getMax$ rule reacts only after all odd numbers were removed from the solution.

$$\langle \ \langle selectEvens, 2, 6, 8 \rangle, \textbf{ replace-one } \langle selectEvens = s, \omega \rangle \textbf{ by } getMax, \ \omega \ \rangle$$

The higher-order rule then extracts remaining numbers, and introduces the $getMax$ rule, so triggering the second phase of the program where the maximum value is kept. The solution is:

$$\langle 2, 6, 8, getMax \rangle$$

$getMax$ then reacts with pairs of integers until only the value 8 remains. Note that, due to the higher-order, putting both rules directly in the integers solution

could result in a wrong result as the pipeline between the two rules would be broken, possibly leading to a wrong result, for instance if $getMax$ reacts first with the 8 and 9, thus deleting the value 8.

2.2 Distributed *Chemical* Coordination Architecture

The architectural framework used in this paper has been described in [9]. It allows several entities to coordinate with each others by reading and writing a shared multiset. It is illustrated in Figure 1. The workflow is represented by a multiset of molecules containing the information (data and control flow) needed to describe and execute it. The execution takes place inside each service reading the chemical solution from the multiset and (re)writing it at the end of each execution's step. In these distributed settings, this multiset is envisioned as a space shared by the services. More precisely, the multiset is composed of as many sub-solutions as there are ChWSes in the workflow, and is of the form:

$$\langle\ ChWS1 : \langle \ldots \rangle,\ ChWS2 : \langle \ldots \rangle,\ \ \ldots,\ ChWSn : \langle \ldots \rangle\ \rangle$$

Invocations of services are themselves encapsulated in a *Chemical Web Service* (ChWS), integrating an HOCL interpreter playing the role of a local workflow (co-)engine. Physically, ChWSes are running on some nodes of the targeted platform, and are logically identified by symbolic names into this multiset ($ChWSi$). In Figure 2, an abstract workflow with several services is translated into a molecular composition. Each ChWS has a library of available generic rules at its disposal. Some of these molecules are composed based on data molecule dependencies, molecules produced by one reaction triggerring other reactions, and so on.

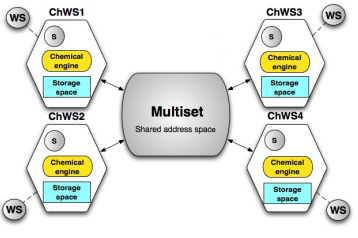


Fig. 1. The proposed architecture

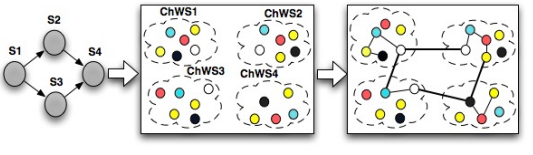


Fig. 2. Molecular composition from a workflow

3 Molecular Compositions

Based on the architectural framework presented in previous section, we now present the contribution of the paper: the expression of an autonomic and decentralized execution of various workflow patterns following an artificial molecular composition process. Molecules, reaction rules and their composition and distribution over services to compose, are presented. We distinguish two types of rules inside the multiset: (1) the rules describing data and control flows of a specific

workflow to be executed, and more importantly here, (2) workflow-independent rules for the coordination of the execution of any workflow referred to as *generic rules*. In the multiset, each sub-solution will only contain the molecules and rules (defined below) required to perform the desired patterns.

3.1 Generic Rules for Invocation and Transfer

Common tasks in a workflow of services are service invocation and information transfer between services. The three common rules responsible for these tasks are illustrated in Algorithm 1. They are present in virtually every existing pattern. The *invokeServ* rule, on reaction, invokes the actual Web Service S_i, by consuming the tuples CALL:*Si* whose presence indicates the potentiality of its invocation, and PARAM:$\langle\ in_1, \ldots, in_n\ \rangle$ representing its input parameters, and generates molecules encapsulating the results. The particular molecule FLAG_INVOKE, used in particular cases illustrated in the following, indicates that the invocation can actually take place. The *preparePass* rule is used to prepare messages containing the results to be transfered. Rule *passInfo*, on reaction, transfers the molecule ω_1 from sub-solution *ChWSi* to sub-solution *ChWSj*.

Algorithm 1. Common generic rules.

2.01 **let** *invokeServ* = **replace** ChWSi:⟨CALL:Si, PARAM:⟨$\text{in}_1, \ldots, \text{in}_n$⟩, FLAG_INVOKE, ω ⟩
2.02 **by** ChWSi:⟨RESULT:ChWSi:⟨value⟩, ω ⟩
2.03 **let** *preparePass* = **replace** ChWSi:⟨RESULT:ChWSi:⟨value⟩, DEST:ChWSj, ω⟩
2.04 **by** ChWSi:⟨PASS:ChWSj:⟨COMPLETED:ChWSi:⟨value⟩ ⟩, ω ⟩
2.05 **let** *passInfo* = **replace** ChWSi:⟨PASS:ChWSj:⟨ ω_1 ⟩, ω_2 ⟩, ChWSj:⟨ ω_3 ⟩
2.06 **by** ChWSi:⟨ ω_2 ⟩, ChWSj:⟨ ω_1, ω_3 ⟩

3.2 Solving Workflow Patterns

We now present the details of the molecular compositions solving complex workflow patterns. A table of the details of all molecules used in the following with their details can be found in the research report [10].

Parallel Split Pattern. A parallel split consists of one single thread splitting into multiple parallel threads (See Figure 3.)
Chemical view: The data to be transferred, is one molecule produced inside the source ChWS sub-solution, and moved to the multiple destination ChWSes. Reactions for these transfers are triggered in parallel (implicitly, following the chemical model), following the *passInfo* rule (detailed before in Algorithm 1), and placed inside the sub-solution of the source ChWS (numbered 1 in Figure 3).

Synchronization Pattern. A Synchronization pattern is a process where multiple (source) parallel branches are merged into one single (destination) thread (See Figure 4).

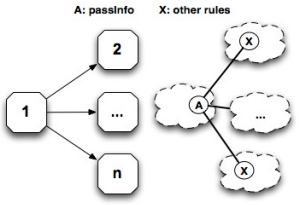


Fig. 3. Parallel split

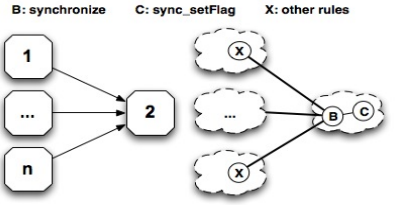


Fig. 4. Synchronization

Chemical view: A synchronization pattern should involve reactions able to gather all incoming result from the source ChWS sub-solutions on the destination ChWS. As detailed in Algorithm 2, the *synchronize* rule allows to gather all the incoming COMPLETED*:ChWSi:⟨value⟩* molecules, specified in the molecule SYNC_SRC*:⟨ ChWSi, ω_1 ⟩* specifying all sources. When all molecules are gathered on the destination ChWS, another reaction, specified by the rule *sync_setFlag*, is triggered to produce a molecule FLAG_INVOKE allowing the destination service (numbered 2 on Figure 4) to be called (Line 3.03).

Algorithm 2. Generic rules - Synchronization.

3.01 **let** *synchronize*=**replace** COMPLETED:ChWSi:⟨value⟩,SYNC_SRC:⟨ChWSi,ω_1 ⟩, SYNC_INBOX:⟨ ω_2⟩
3.02 **by** SYNC_INBOX:⟨ COMPLETED:ChWSi:⟨value⟩, ω_2 ⟩, SYNC_SRC:⟨ ω_1 ⟩
3.03 **let** *sync_setFlag* = **replace-one** SYNC_SRC:⟨ ⟩ **by** FLAG_INVOKE

Discriminator Pattern. In a discriminator pattern, a service is activated only once by its first completed incoming branch, ignoring any further completion of incoming branches. (See Figure 5).
Chemical view: Each source ChWS prepares a special message including a molecule indicating that discrimination is needed. As detailed in Algorithm 3, the *discr_preparePass* reaction rule, on every source ChWS, adds a DISCRIMINATOR molecule to the information sent (Lines 4.01 and 4.02). The destination ChWS waits for the DISCRIMINATOR molecule and only the first received will react with the one-shot *discr_setFlag* rule, setting the invocation flag (Line 4.03), and thus leading to the ignorance of subsequent incoming DISCRIMINATOR molecules.

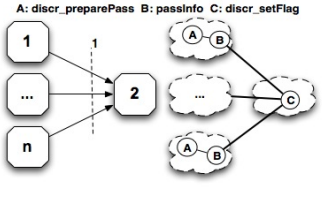


Fig. 5. Discriminator

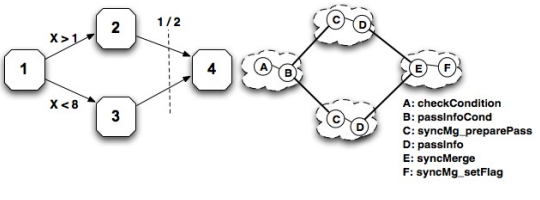


Fig. 6. Synchronization merge

Algorithm 3. Generic rules - Discriminator.

```
4.01 let discr_prepare_Pass = replace DEST:ChWSj, RESULT:ChWSi:⟨value⟩
4.02                          by PASS:ChWSj:⟨ COMPLETED:ChWSi:⟨value⟩, DISCRIMINATOR ⟩
4.03 let discr_setFlag = replace-one DISCRIMINATOR by FLAG_INVOKE
```

Synchronization Merge Pattern. In a synchronization merge pattern, a service (service 4 on Figure 6) synchronizes several of its incoming branches. The number of branches that the service has to wait for is defined at runtime, depending on the result of a previous service (service 1 on Figure 6).
Chemical view: As detailed in Algorithm 4, the pattern is achieved by transferring one molecule SYNCMG_SRC:⟨ *ChWSi,* ω ⟩ representing all the ChWSes from which one molecule of the form COMPLETED:*ChWSi:*⟨*value*⟩ has to be received in the destination ChWS. This molecule corresponds to all incoming branches that need to be activated. Following Figure 6, this is achieved by the molecule A in Service 1. The multi-choice is then executed on Service 1, actually activating one or both services 2 and 3, through the *passInfoCond* rule. The *syncMerge* rule then waits for the required molecules, and finally the *syncMg_setFlag* rule is triggered, producing a new molecule FLAG_INVOKE, allowing for the invocation. The SYNCMG_INBOX:⟨ ω ⟩ molecule stores the already received COMPLETED:*ChWSi:*⟨*value*⟩ molecules.

Algorithm 4. Generic rules - Synchronization merge.

```
5.01 let syncMg_prepare_Pass = replace DEST:ChWSj, RESULT:ChWSi:⟨value⟩,
5.02                                   SYNCMG_SRC:⟨ChWSi, ω ⟩
5.03                         by PASS:ChWSj:⟨ COMPLETED:ChWSi:⟨value⟩, SYNCMG_SRC:⟨ ChWSi, ω ⟩ ⟩
5.04 let syncMerge = replace COMPLETED:ChWSi:⟨value⟩,
5.05                         SYNCMG_SRC:⟨ ChWSi, ω1 ⟩ , SYNCMG_INBOX:⟨ ω2 ⟩
5.06                         by SYNCMG_INBOX:⟨COMPLETED:ChWSi:⟨value⟩, ω2 ⟩, SYNCMG_SRC:⟨ ω1 ⟩
5.07 let syncMg_setFlag = replace-one SYNCMG_SRC:⟨ ⟩ by FLAG_INVOKE
```

To sum up, coordination responsibilities are distributed as reactions take place where molecules and rules are set. Other classical workflow patterns identified by Van der Aalst *et al.* in [19] such as *exclusive choice*, *multi-choice*, *simple merge*, *multi-merge*, or cancellation, can be treated similarly. We omit their description for space reasons. Their description can be found in the research report [10].

4 Experimental Evaluation

To establish a proof of concept on *molecular service composition*, we developed a software prototype, and deployed it over the nation-wide platform Grid'5000 [1], connecting together eight geographically-distributed federations of clusters.

4.1 Software Prototype

The prototype is written in Java, and relies on the *on-the-fly* compilation of HOCL programs. For the sake of comparison, and to highlight the advantages and drawbacks of our solution, two versions have been deployed. The first one, referred to as the *centralized* version, gathers all elements (service invocations and multiset) on a single physical machine. The second one, *decentralized*, deploys each ChWS, and the multiset on different machines, interconnected by the network. We now describe both versions in more details.

Centralized prototype. The centralized version presents a unique ChWS, a *chemical workflow service*, acting as a coordinator node, internally composed of a multiset playing the role of storage space and a workflow engine, as shown on Figure 7. Therefore, given a workflow definition, the unique ChWS executes the workflow by reading and writing the multiset. The interface between the chemical engine and the distant services themselves was implemented with the *service caller*, relying on the DAIOS framework [13] developed at Technische Universität Wien, which provides an abstraction layer allowing dynamic and transparent connections to different flavours of services (SOAP or RESTFul).

Decentralized prototype. The decentralized architecture differs in the sense that the multiset is now a shared space playing the role of a communication mechanism as well as a storage system. On each ChWS, a local storage space acts as a temporary container for its corresponding sub-solution to be processed by the local HOCL interpreter. As shown by Figure 8, ChWSes communicate with the multiset through the Java Message Service publisher/subscriber communication model. When the multiset detects changes in the sub-solution of a particular ChWS, it sends the new content to this ChWS. The sub-solution received is then processed, modified by the local HOCL interpreter and then published to the multiset for update. Each ChWS is now interfaced to one concrete WS, through the *service caller*.

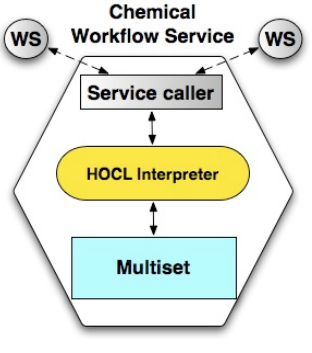


Fig. 7. Centralized prototype

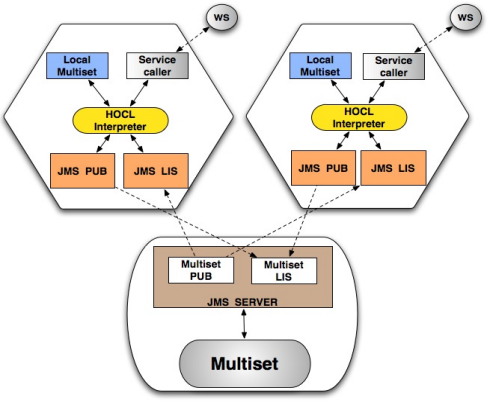


Fig. 8. Decentralized prototype

4.2 Results

We now present the performance of our approach with both propotype versions, using *synchronization*, *parallel split*, *discriminator* and *synchronization merge* patterns. These patterns can be extended in terms of number of services to obtain significant results regarding scalability.

Each pattern was deployed with 5, 15, 30, 45 and 60 tasks, where tasks are web services to be executed. Each node, in the decentralized prototype, is deployed on a distinct Grid'5000 node. Each experiment has been repeated 6 times. Charts show the results averaged on the 6 experiments. The *synchronisation* pattern consisted in one service realizing the synchronization, the others being its incoming branches. The *parallel split* consisted in one source node, the others being its outgoing branches. Similar *vertical* extensions have been done for the *discriminator* and *synchronization merge* patterns.

In Figure 9, performance obtained with the *synchronization* and *parallel split* patterns are given. A first observation is that decentralizing the process brings a non-negligible performance improvement, especially when the number of tasks to coordinate increases, the centralized version suffering from the concentrated workload on the unique coordinator for all the branches.

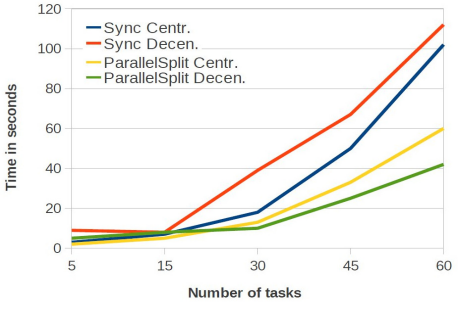


Fig. 9. Performance, basic patterns

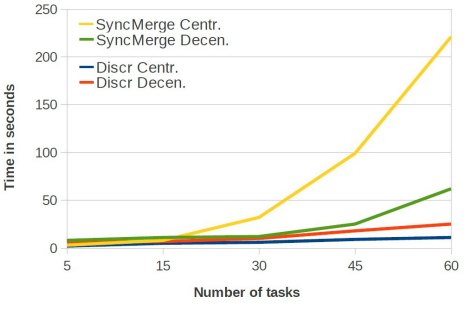


Fig. 10. Performance, advanced patterns

Next, we considered the more complex branching and merging concepts used in the *discriminator* and *synchronization merge* patterns. Results are shown in Figure 10. A first encouraging result is that the execution time for the *discriminator* shows similar performance evolution for both versions. The *synchronization merge* pattern highlights again the relevance of a decentralized approach, as the performance degradation in a centralized environment is similarly experienced. Again, this can be explained by the complexity of the pattern, composed of a *multi-choice* and a *synchronization* pattern, leading to a severe increase in the coordination's workload.

This series of experiments provides a proof of concept of the *chemical* service composition, and experimentally highlights the added value of a decentralized execution models to deal with the execution of large workflows.

5 Related Works

Early workflow executable languages, such as BPEL [16], YAWL [2], or other proprietary languages [14], lack of means to express dynamic behaviors. More recently, alternative approaches were proposed providing this dynamic nature to the service composition. A system governed and guided by rules mechanisms supporting dynamic binding and flexible service composition was proposed in [12]. In [7], a BPEL extension based on aspect oriented programming supports the dynamic adaptation of composition at runtime. Other approaches, such as [21], propose coordination models based on data-driven languages like Linda [11], to facilitate dynamic service matching and service composition. However, these approaches rely on architectures where the composition is still managed by a central coordinator node. In the same vein, works such as [15], propose a distributed architecture based on Linda where distributed tuple spaces store data and programs as tuples, allowing for mobile computations by transferring programs from one tuple to another. While our work presents similarities with this last approach, the chemical paradigm allows an increased abstraction level while providing support for dynamics.

A more recent series of works addresses the need for decentralization in workflow execution. The idea they promote is to replace a centralized BPEL engine by a set of distributed, loosely coupled, cooperating nodes [6, 17]. These works propose a system based on workflow management components: each workflow component contains sufficient information so that they can be managed by local nodes rather than one central coordinator. Again, this work present architectural similarities with our approach, but the chemical model allows for more expressive abstractions. Some languages have also been proposed for providing a distributed support to service coordination [18]. However, they are finally turned into BPEL for the execution, losing accuracy and expressiveness in the translation.

6 Conclusion

Although the design and implementation of workflow management systems is a subject of considerable research, most recent solutions are still mostly based on a centralized coordination model. Likewise, the workflow executable languages used in these systems are intrinsically static and do not provide concepts for autonomous and decentralized workflow execution. Thus, we need to promote a decentralized workflow execution systems, based on executable language allowing to partition composite web services integrating complex patterns into fragments without losing information.

In this paper, we have proposed concepts for a chemistry-inspired autonomic workflow execution, namely *molecular composition*. Such an analogy was shown to be able to express the decentralized and autonomous execution of a wide variety of workflow patterns. A ready-to-use HOCL library for this purpose has been proposed, and implemented in a middleware prototype. Its experimentation over a distribute platform has been conducted, providing a proof of concept and suggesting promising performance.

References

1. Grid'5000 (June 2011), http://www.grid5000.fr
2. van der Aalst, W.M.P., ter Hofstede, A.H.M.: YAWL: yet another workflow language. Information Systems 30(4), 245–275 (2005)
3. Alonso, G., Mohan, C., Agrawal, D., Abbadi, A.E.: Functionality and limitations of current workflow management systems. IEEE Expert 12 (1997)
4. Banâtre, J., Fradet, P., Radenac, Y.: Generalised multisets for chemical programming. Mathematical Structures in Computer Science 16(4), 557–580 (2006)
5. Banâtre, J.P., Priol, T., Radenac, Y.: Chemical Programming of Future Service-oriented Architectures. Journal of Software 4(7), 738–746 (2009)
6. Buhler, P.A., Vidal, J.M.: Enacting BPEL4WS specified workflows with multiagent systems. In: Proceedings of the Workshop on WSABE (2004)
7. Charfi, A., Mezini, M.: AO4BPEL: an aspect-oriented extension to BPEL. World Wide Web 10, 309–344 (2007)
8. Dittrich, P., Ziegler, J., Banzhaf, W.: Artificial chemistries – a Review. Artificial Life 7, 225–275 (2001)
9. Fernández, H., Priol, T., Tedeschi, C.: Decentralized Approach for Execution of Composite Web Services using the Chemical Paradigm. In: 8th International Conference on Web Services (ICWS 2010), pp. 139–146. IEEE, Miami (2010)
10. Fernández, H., Tedeschi, C., Priol, T.: Self-coordination of Workflow Execution Through Molecular Composition. Research Report RR-7610, INRIA (May 2011), http://hal.inria.fr/inria-00590357/PDF/RR-7610.pdf
11. Gelernter, D., Carriero, N.: Coordination languages and their significance. Commun. ACM 35(2), 96–107 (1992)
12. Laliwala, Z., Khosla, R., Majumdar, P., Chaudhary, S.: Semantic and rules based Event-Driven dynamic web services composition for automation of business processes. In: Services Computing Workshops, SCW 2006, pp. 175–182 (2006)
13. Leitner, P., Rosenberg, F., Dustdar, S.: Daios: Efficient dynamic web service invocation. IEEE Internet Computing 13(3), 72–80 (2009)
14. Montagut, F., Molva, R.: Enabling pervasive execution of workflows. In: International Conference on Collaborative Computing: Networking, Applications and Worksharing, p. 10 (2005)
15. Nicola, R.D., Ferrari, G., Pugliese, R.: KLAIM: a kernel language for agents interaction and mobility. IEEE Transactions On Software Engineering 24 (1997)
16. OASIS: Web services business process execution language, (WS-BPEL 2.0) (2007)
17. Sonntag, M., Gorlach, K., Karastoyanova, D., Leymann, F., Reiter, M.: Process space-based scientific workflow enactment. International Journal of Business Process Integration and Management 5(1), 32–44 (2010)
18. Taylor, I.J., Deelman, E., Gannon, D.B., Shields, M., Slominski, A.: Adapting BPEL to scientific workflows. In: Workflows for e-Science, pp. 208–226 (2007)
19. Van Der Aalst, W., Ter Hofstede, A., Kiepuszewski, B., Barros, A.: Workflow patterns. Distrib. Parallel Databases 14(1), 5–51 (2003)
20. Viroli, M., Zambonelli, F.: A biochemical approach to adaptive service ecosystems. Information Sciences, 1–17 (2009)
21. Wutke, D., Martin, D., Leymann, F.: Facilitating Complex Web Service Interactions through a Tuplespace Binding. In: Meier, R., Terzis, S. (eds.) DAIS 2008. LNCS, vol. 5053, pp. 275–280. Springer, Heidelberg (2008)

Using a Lifecycle Model for Developing and Executing Real-Time Online Applications on Clouds

Dominik Meiländer[1], Antonio Bucchiarone[2], Cinzia Cappiello[3], Elisabetta Di Nitto[3], and Sergei Gorlatch[1]

[1] University of Muenster, Germany
[2] Fondazione Bruno Kessler, Italy
[3] Politecnico di Milano, Italy
d.meil@uni-muenster.de

Abstract. We describe how the generic Lifecycle Model developed in the S-Cube project for the design and management of service-based applications (SBA) can be utilized in the context of Cloud Computing. In particular, we focus on the fact that the Infrastructure-as-a-Service approach enables the development of Real-Time Online Interactive Applications (ROIA), which include multi-player online computer games, interactive e-learning and training applications and high-performance simulations in virtual environments. We illustrate how the Lifecycle Model expresses the major design and execution aspects of ROIA on Clouds by addressing the specific characteristics of ROIA: a large number of concurrent users connected to a single application instance, enforcement of Quality of Service (QoS) parameters, adaptivity to changing loads, and frequent real-time interactions between users and services. We describe how our novel resource management system RTF-RMS implements concrete mechanisms that support the developer in designing adaptable ROIA on Clouds according to the different phases of the Lifecycle Model. Our experimental results demonstrate the influence of the proposed adaptation mechanisms on the application performance.

1 Introduction

Service-oriented applications are developed for constantly changing environments with the expectation that they will evolve over time. Several service-oriented system engineering (SOSE) methodologies have been proposed aiming at providing methods and (sometimes) tools for researchers and practitioners to engineer service-oriented systems. SOSE methodologies are more complex than traditional software engineering (TSE) methodologies: the additional complexity results mainly from open world assumptions, co-existence of many stakeholders with conflicting requirements and the demand for adaptable systems. A number of service lifecycle models have been proposed by both industry and academia. However, none of the proposed models has either reached a sufficient level of maturity or been able to fully express the aspects peculiar to SOSE. Within the S-Cube project [1] a new Lifecycle Model was designed that combines existing techniques and methodologies from TSE and SOSE to improve the process through which service-based applications will be developed.

This paper extends our previous work [2] on studying how the S-Cube Lifecycle Model can be applied on the emerging and challenging domain of *Real-Time Online Interactive Applications (ROIA)* including multi-player online games, high-performance

G. Pallis et al. (Eds.): ICSOC 2011, LNCS 7221, pp. 33–43, 2012.

simulations, e-learning applications, etc. In particular, we study how to use server resources economically, which is difficult due to continuously changing user numbers.

Cloud Computing with its *Infrastructure-as-a-Service (IaaS)* approach offers new opportunities for ROIA execution and promises a potentially unlimited scalability by distributing application processing on an arbitrary number of resources given suitable adaptation mechanisms. Clouds allow for adding/removing resources on demand. This opens for ROIA an opportunity to serve very high numbers of users and still comply with QoS demands. Despite a variable number of users, Cloud resources can be used efficiently if the application supports adding/removing resources during runtime. Hence, using Cloud Computing for resource provision and the Lifecycle model for implementing adaptable ROIA complement each other.

This paper studies how Cloud Computing and the S-Cube Lifecycle Model can be utilized for ROIA applications. We illustrate how the Lifecycle Model expresses the major design and execution aspects of ROIA on Clouds and present our novel resource management system RTF-RMS that implements concrete mechanisms for ROIA development and adaptation according to the Lifecycle. We report experimental results on the influence of the proposed adaptation mechanisms on the application performance.

The paper is structured as follows. We introduce the S-Cube Lifecycle Model in Section 2, followed by a description of the challenges in ROIA development and execution on Clouds in Section 3. Section 4 illustrates how the Lifecycle Model is applied for ROIA development on Clouds using RTF-RMS. Section 5 reports experimental results on the adaptation of a sample ROIA, and Section 6 concludes the paper.

2 Lifecycle Model

Many of the existing development methodologies for service-based applications (SBA) are based on the results carried out in the fields of classical software and system engineering and do not facilitate SBA adaptation [3,4,5]. Some of the reported SBA development approaches such as SOUP (Service Oriented Unified Process) [6] or the approach by Linner et al [7] do support some level of adaptation, however, they lack sufficient process details. Lane and Richardson [8] carried out a systematic literature review of SBA development approaches, they identified 57 such approaches of which there were only eight that specifically dealt with adaptation. Only four of these eight approaches target the adaptation of SBAs, the others target the adaptation of services.

Each of the four approaches shows interesting features, but even those that enable the definition of various adaptation strategies lack a coherent approach to support designers in this complex task. Moreover, they focus on the implementation process without considering what impact adaptation has on the rest of the development and operational lifecycle [9,10,11]. Finally, they also tend to focus on particular types of adaptation, such as adaptation due to requirement violations [12], or service substitution due to application constraints [13], so it is difficult to elicit generic adaptation mechanisms from them. In summary, each of these approaches focused on the analysis and design processes without consideration for any other development or runtime processes.

The Lifecycle Model proposed in the S-Cube project (see Fig. 1) aims to support the design of adaptable SBAs [14]. It provides a solid reference [15] for practitioners who are planning to develop adaptable SBAs since that it has advantages over similar

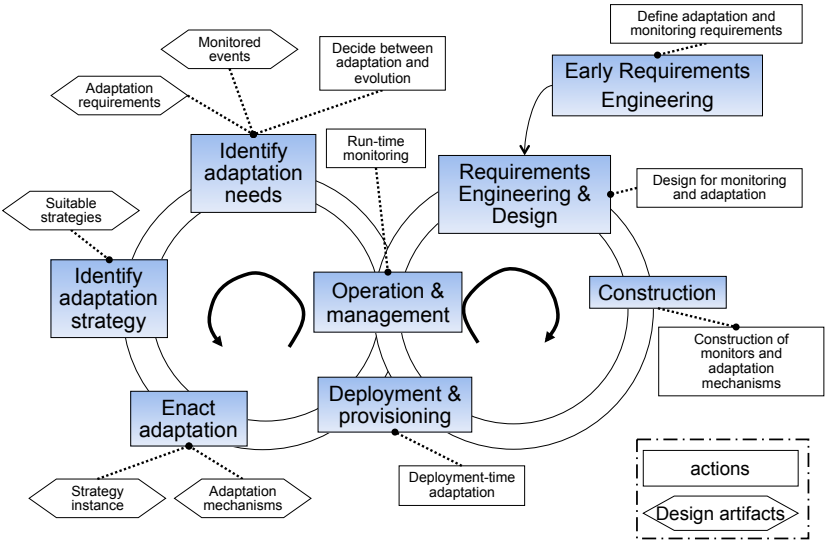


Fig. 1. Lifecycle for adaptable service-oriented systems

approaches in that it focuses on software process rather than the specific adaptation mechanism implementation techniques. It highlights not only the typical design-time iteration cycle, but it also introduces a new iteration cycle that is performed at runtime when the application needs to be adapted on-the-fly. Both cycles coexist and support each other during the lifetime of the application. In particular, the design time activities allow for *evolution* of the application, i.e., introduction of permanent and, usually, important changes, while the runtime activities allow for temporary *adaptation* of the application to a specific situation. At design time, it is important to analyze functional and non-functional requirements, context and machine characteristics in order to (i) identify the types of changes that trigger self-adaptation, (ii) define mechanisms to monitor the environment and the system behavior, and (iii) develop strategies for self-adaptation.

In the *(Early) Requirement Engineering and Design* phase, the relevant context dimensions and the application and system characteristics are considered in order to elicitate adaptation and monitoring requirements and in particular define the types of changes that trigger self-adaptation. Subsequently, during the *Construction* phase, the corresponding monitoring and adaptation mechanisms are designed, developed and then refined until the *Deployment and Provisioning* phase. At runtime (*Operation and Management* phase), the system is continuously monitored in order to support the detection of the relevant context and system changes. When changes occur, the system might require evolution or adaptation interventions. Evolution is performed if the system needs to be redesigned and thus it requires the reiteration of the described cycle starting from the requirements engineering and design phase.

In the *Identify adaptation need* phase, specific situations that demand for adaptation are identified. Each adaptation need has to be associated with particular *Adaptation Strategies* that are able to satisfy the corresponding adaptation requirements. Based on

the current situation, the knowledge obtained from previous executions, and the available adaptation strategies, a reasoner (e.g., a resource management system) selects the most suitable adaptation strategy that will be performed in the *Enact adaptation* phase. Fig. 1 highlights for each phase the various adaptation- and monitoring-specific actions (boxes) carried out throughout the lifetime of an SBA and the main design artifacts that are exploited to perform adaptation (hexagons).

3 ROIA Development and Execution on Clouds

In Real-Time Online Interactive Applications (ROIA), there are typically multiple users who concurrently access a common application state and interact with each other within one virtual environment. The users access the application from different client machines and control their avatars that can influence other users' avatars. Since ROIA have very high performance requirements, the application state processing is performed on multiple servers: the virtual environment is divided into disjoint areas (*zones*) and each server is processing clients inside a particular zone, i.e., the overall workload of the application is distributed on multiple resources. Hence, ROIA are highly distributed applications with challenging QoS demands, such as: short response times (about 0.1-1.5 s), high update rate of the application (up to 50 Hz), large and frequently changing number of users in a single application instance (up to 10^4 simultaneously).

An important challenge is that the number of users participating in a ROIA session and, thus, the workload, is often subject to daytime-dependent changes. As a consequence, expensive up-front investments are made to build a server pool which is able to handle peak user numbers but will be underutilized most of the time when the load is below the peak. Hence, dynamic adaptation of ROIA sessions during runtime is crucial.

We address this challenge by using Cloud Computing for resource provision. Cloud Computing allows to add/remove resources on demand and promises a potentially unlimited scalability by distributing frequent state computations on an arbitrary number of resources given suitable adaptation mechanisms. Despite a variable number of users, Cloud resources can be used efficiently if the application provides suitable adaptation mechanisms. Hence, using Cloud Computing for resource provision and the Lifecycle Model for implementing adaptable ROIA complement each other.

In order to support ROIA development and adaptation on Clouds, we develop the RTF-RMS resource management system [16] on top of the Real-Time Framework (RTF) [17]. RTF-RMS implements the following mechanisms for ROIA development on Clouds:

1. *Monitoring* of application-specific data, e.g., update rate, number of entities, etc.
2. *Distribution handling* for the dynamic adaptation of application sessions by adding/removing Cloud resources using particular adaptation strategies (described below).
3. *Application profiles* for implementation of application-specific adaptation triggers.
4. High-level *development tools* for communication and application state distribution.

4 Using the Lifecycle Model for ROIA Development on Clouds

This section describes how RTF-RMS supports the developer in designing adaptable ROIA according to the different phases of the Lifecycle Model. In [2], we showed how

the Lifecycle described in Section 2 can be applied for ROIA development. In this section, we demonstrate how the Lifecycle Model is used for ROIA development in Cloud environments by exploiting RTF-RMS.

The design of an adaptive application requires the definition of suitable adaptation strategies and adaptation triggers. In the following, we illustrate how RTF-RMS supports the developer in designing adaptable ROIA in Cloud environments according to the different phases of the Lifecycle model.

In the "Requirement Engineering" phase, the application developer must identify suitable adaptation requirements for his application. For ROIA, the mechanisms for monitoring and adaptation should be non-intrusive, i.e., take place in parallel with the application execution, such that users are not aware of changes inside the application.

For the "Construction" phase, RTF-RMS provides the developer with a C++ library of high-level functions for optimized communication handling (client-server and inter-server) and efficient application state distribution in a multi-server environment. By using RTF-RMS for communication and distribution handling, monitoring mechanisms are automatically integrated in the application processing. These monitoring mechanisms are used in the next phase of the Lifecycle to implement adaptation triggers.

In the "Deployment and Provisioning" phase, trigger rules for each adaptation trigger are defined. We describe the implementation of adaptation triggers in Section 4.1.

In the "Operation and Management" phase, the application is running and monitoring data are checked continuously against the trigger rules to detect changes in the context or in the system that could require adaptation.

In the "Identify adaptation need" phase, RTF-RMS detects a violation of trigger rules which indicates the demand for adaptation.

In the "Identify adaptation strategy" phase, RTF-RMS analyzes the number of application servers and their current workload to choose an adaptation strategy. A detailed description of adaptation strategies provided by RTF-RMS is given in Section 4.2.

In the "Enact adaptation" phase, RTF-RMS enacts the chosen adaptation strategy and changes the distribution of the application processing accordingly.

4.1 Adaptation Triggers for ROIA on Clouds

In general, the adaptation is triggered when some changes occurs. Such changes may affect the component services or the context of the considered application. ROIA require an adaptation when one of the following changes occurs:

1. *Change in Quality of Service*: QoS violations may be caused by unreliable Cloud resources. QoS violations for ROIA may be related to changes in update rate, response time, throughput, resource usage, packet loss and service availability.
2. *Change in computational context*: application requirements sometimes change on the basis of variations of the computational context such as variations of CPU and memory load or incoming/outgoing bandwidth.
3. *Change in business context*: it refers to changes in user preferences which were not predicted in advance, e.g., too many concurrent users connected to the application or the increase of the number of requests per application.

RTF-RMS provides a generic mechanism to implement adaptation triggers by specifying an *application profile*. In the application profile, developers can specify significant

Listing 1. Excerpt from an example application profile for a fast-paced action game

```
<appProfile>
  <metric>UpdateRate</metric>
  <addResourceThreshold>25</addResourceThreshold>
  <removeResourceThreshold>100</removeResourceThreshold>
</appProfile>
```

monitoring values for their particular application, e.g., update rate in Hz, see Listing 1 for an example. For each monitoring value in the application profile thresholds have to be defined. If a monitoring value exceeds the `addResourceThreshold`, a new resource is added to the application processing; if monitoring values of any application server fall below the `removeResourceThreshold` dispensable resources are removed. Application developers can find suitable values for these thresholds by considering the runtime behaviour of their application on physical resources and calculating the virtualization overhead of Cloud resources, or by conducting benchmark experiments in a Cloud.

In order to choose between different adaptation strategies, RTF-RMS creates *reports* for each application server. A report describes the load of a particular server, e.g., current update rate in Hz. A *zone report* is created for each zone by collecting reports from all servers involved in the zone processing and calculating their average load.

Fig. 2 illustrates an example of choosing adaptation strategies using zone reports. We assigned Server A to the processing of Zone 1, and Server B and C were assigned to Zone 2. The zone report of Zone 1 identifies an average update rate of 20 Hz. Given an `addResourceThreshold` of 25 Hz, RTF-RMS decides to add a new resource to the processing of Zone 1. The zone report of Zone 2 identifies an average update rate of 60 Hz which is between `addResourceThreshold` and `removeResourceThreshold`, but Server B has an update rate of 20 Hz. Hence, RTF-RMS migrates users from Server B to Server C to distribute the workload equally on both servers.

Another challenge for the implementation of adaptation triggers on Clouds is the compensation of long startup time of Cloud resources which may take up to several minutes. In RTF-RMS, multiple requests for new Cloud resources are sent in parallel to the Cloud API in order to start multiple resources as quickly as possible. Moreover, RTF-RMS introduces a *resource buffer* to which a predefined number of Cloud resources are moved in advance, i.e. before they are demanded by the application. Resources in the resource buffer are immediately available. If any resource from the resource buffer is integrated in the application processing, RTF-RMS checks whether new Cloud resources must be started in order to keep a certain number of resources in the resource buffer. A detailed description of how to choose the resource buffer size in order to minimize the cost-overhead generated by leasing resources in advance is provided in [16].

4.2 Adaptation Strategies for ROIA on Clouds

The adaptation triggers described in the previous section identify the need for adaptation which is implemented by different adaptation strategies. In order to react to changes and avoid inefficiencies, it is necessary to identify the most suitable adaptation strategy that is able to align the application behaviour with the context and system requirements.

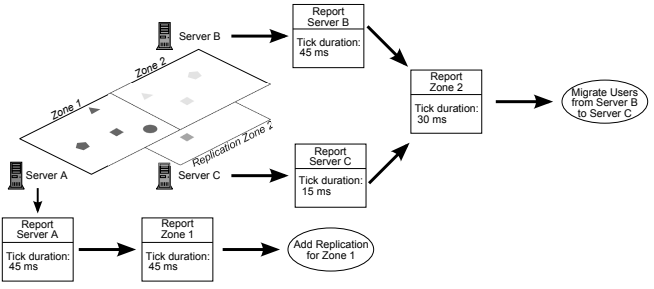


Fig. 2. Finding a suitable adaptation strategy using zone reports

Fig. 3 illustrates how RTF-RMS chooses between four adaptation strategies proposed in our previous work [18]:

1. *User migration*: Users are migrated from an overloaded server to an underutilized server which is replicating the same zone. For this purpose, user connections are switched from one server to another. RTF-RMS distributes users by default equally between the application servers for a particular zone. User migration is restricted to servers that are replicating the same zone because managing users located in different zones on the same server would considerably increase the inter-server communication for processing user interactions which are typically limited to nearby users. User migration is the preferred action if the load of an overloaded server can be compensated by running resources.
2. *Replication enactment*: New application servers are added in order to provide more computation power to the highly frequented zone. This strategy is called replication: each application server keeps a complete copy of the application state, but each server is responsible for computing a disjoint subset of entities. As soon as the new server has been added, RTF-RMS migrates a number of users to the new replica in order to balance the load. Replication implies an additional inter-server communication, so its scalability is limited. Since the replication overhead depends on the inter-server communication in a particular application, the maximum number of replicas per zone can be specified in the application profile. If the number of active replicas for a particular zone is below the maximum number of replicas specified by the application profile, then replication is used; otherwise the resource substitution strategy (described next) is preferred.
3. *Resource substitution*: An existing application server is substituted by a more powerful resource in order to increase the computation power for highly frequented zones. For this purpose, RTF-RMS replicates the targeted zone on the new resource and migrates all clients to the new server. The substituted server is shut down.

   If no better resources are available for substitution, the application reached a critical user density, i.e., large numbers of users in the same geographical area, which cannot be further improved by the generic adaptation strategies offered by RTF-RMS. In this case, the application requires redesign according to the design time activities of the Lifecycle Model.

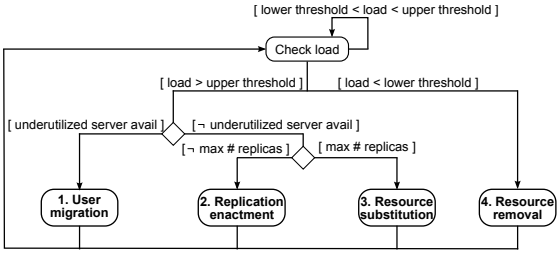


Fig. 3. RTF-RMS chooses between four different adaptation strategies

4. *Resource removal*: If the update rate of an underutilized application server falls below the `removeResourceThreshold`, RTF-RMS checks whether the zone that is managed by this server is replicated by other servers. If not, nothing happens since each zone must be assigned to at least one application server. If other application servers are replicating this zone, users are migrated equally to these servers, after which the underutilized server is shut down.

Another challenge for the cost-efficient adaptation of ROIA on Clouds is the consideration of leasing periods. In commercial Cloud systems, resources are typically leased and paid per hour or some longer leasing period. Since ROIA have dynamically changing user numbers, the time after which Cloud resources become dispensable is very variable. However, resources will not be used cost-efficiently if they are shut down before the end of their leasing period. In RTF-RMS, resources that have become dispensable are removed from the application processing and moved to the resource buffer. Cloud resources in the buffer are shut down at the end of their leasing period or they are integrated in the application processing again if required.

5 Experiments

In the following, we target the "Enact adaptation" phase of the Lifecycle Model and report experimental results of the replication enactment adaptation strategy using an example of a multi-player action game called RTFDemo [17]. In our experiments, we evaluate how RTF-RMS triggers adaptation if QoS changes as described in Section 4.1; from the adaptation triggers described in Section 4.1, we chose the update rate to trigger adaptation. In order to provide a seamless gaming experience, users should not receive less than 25 updates per second over a longer time period. Hence, we defined an adaptation trigger rule with 25 updates per second as the `addResourceThreshold`.

In our experiments, we use a private Cloud with the Eucalyptus framework (version 2.0.2) [19]; servers are Intel Core Duo PCs with 2.66 GHz and 2 GB of RAM.

We start a single RTFDemo server on a Cloud resource and connect 260 clients to it. Fig. 4 shows that the update rate of Server 1 initially drops with the growing number of connected clients. When the update rate of Server 1 falls below the threshold of 25 Hz, RTF-RMS requests a new Cloud resource from the resource buffer (for this experiment, the size of the buffer was configured as 1). Although the requested resource is already

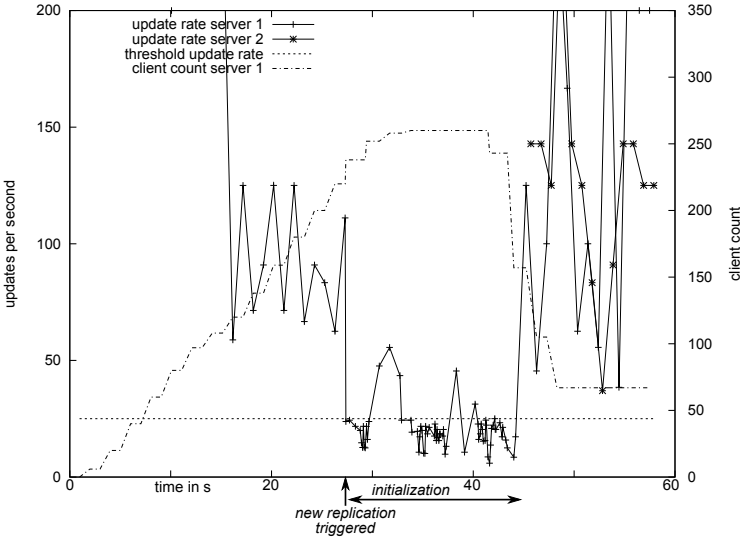


Fig. 4. Load balancing by replication enactment

started up (since it is in the buffer), we observe that a certain time period is still required to add the new server to the application processing and start user migration. This delay of approximately 15 seconds is caused by the initialization and inter-server communication that are required to integrate the new server in the application processing. After the migration is accomplished, the update rate of Server 1 has increased from 20 Hz to about 100 Hz. The update rate of server 1 and server 2 fluctuates between 50 and 200 Hz caused by the continuously changing number of interactions between the 260 clients. Note that if the resource were not in the buffer and would have been started up from scratch, the delay would be much longer, in the order of 130 seconds. Hence, using the resource buffer for starting Cloud resources in advance reduces startup times by a factor of approximately 9 which allows for faster adaptation enactment in contrast to solutions that start resources from scratch, e.g., Amazon Elastic Load Balancing [20].

6 Conclusion

This paper presents how the S-Cube Lifecycle Model can be utilized for developing adaptive ROIA on emerging Cloud systems. We showed how our resource management system RTF-RMS implements concrete mechanisms for ROIA development and execution on Clouds according to the Lifecycle. In extension of our previous work that proved the feasibility of applying the S-Cube Lifecycle on ROIA development on a static set of physical resources [2], this paper targets the specific challenges related to Clouds. In particular, we illustrated how the Cloud influences the definition of adaptation triggers and strategies and how RTF-RMS allows for a cost-effective leasing of Cloud resources on demand by buffering unused resources; thereby the startup times of Cloud resources are reduced. Our adaptation triggers are based on application-specific monitoring values

provided by RTF-RMS, and, hence, go beyond the state-of-the-art adaptation mechanisms on common Cloud platforms that are based on generic system information. Our experimental results demonstrate how the replication enactment adaptation strategy implemented in RTF-RMS improves the performance of a multi-player, real-time online game and how the resource buffer decreases startup times of Cloud resources.

Acknowledgments. We are grateful to the anonymous reviewers for their helpful remarks on the preliminary version of the paper. Our research has received funding from the EC's 7th Framework Programme under grant agreement 215483 (S-Cube).

References

1. The S-Cube project (2011), http://www.s-cube-network.eu
2. Meiländer, D., Gorlatch, S., Cappiello, C., Mazza, V., Kazhamiakin, R., Bucchiarone, A.: Using a Lifecycle Model for Developing and Executing Adaptable Interactive Distributed Applications. In: Di Nitto, E., Yahyapour, R. (eds.) ServiceWave 2010. LNCS, vol. 6481, pp. 175–186. Springer, Heidelberg (2010)
3. Rational, Rational unified process - best practices for software development teams. Tech. Rep. TP026B (1998)
4. Papazoglou, M.P., van den Heuvel, W.: Service-oriented design and development methodology. Int. J. Web Eng. Technol. 2(4), 412–442 (2006)
5. Arsanjani, A., Ghosh, S., Allam, A., Abdollah, T., Gariapathy, S., Holley, K.: SOMA: a method for developing service-oriented solutions. IBM Syst. J. 47, 377–396 (2008)
6. Mittal, K.: Service oriented unified process (2009), http://www.kunalmittal.com/html/soup.html
7. Linner, D., Pfeffer, H., Radusch, I., Steglich, S.: Biology as Inspiration Towards a Novel Service Life-Cycle. In: Xiao, B., Yang, L.T., Ma, J., Muller-Schloer, C., Hua, Y. (eds.) ATC 2007. LNCS, vol. 4610, pp. 94–102. Springer, Heidelberg (2007)
8. Lane, S., Richardson, I.: Process models for service based applications: A systematic literature review. Information and Software Technology (2010)
9. Wautelet, Y., Achbany, Y., Lange, J.-C., Kolp, M.: A Process for Developing Adaptable and Open Service Systems: Application in Supply Chain Management. In: Filipe, J., Cordeiro, J. (eds.) ICEIS 2009. LNBIP, vol. 24, pp. 564–576. Springer, Heidelberg (2009)
10. Vale, S., Hammoudi, S.: Model driven development of context-aware service oriented architecture. In: The 11th IEEE International Conference on Computational Science and Engineering - Workshops, pp. 412–418 (2008)
11. Margaria, T., Steffen, B., Wirsing, M., et al.: SENSORIA Patterns: Augmenting Service Engineering with Formal Analysis, Transformation and Dynamicity. In: Margaria, T., Steffen, B. (eds.) ISoLA 2008. CCIS, vol. 17, pp. 170–190. Springer, Heidelberg (2008)
12. Spanoudakis, G., Zisman, A., Kozlenkov, A.: A service discovery framework for service centric systems. In: 2005 IEEE International Conference on Services Computing, vol. 1, pp. 251–259 (2005)
13. Verma, K., Gomadam, K., Sheth, A.P., Miller, J.A., Wu, Z.: The METEOR-S approach for configuring and executing dynamic web processes. Tech. rep. (2005)
14. Bucchiarone, A., Cappiello, C., Di Nitto, E., Kazhamiakin, R., Mazza, V., Pistore, M.: Design for Adaptation of Service-Based Applications: Main Issues and Requirements. In: Dan, A., Gittler, F., Toumani, F. (eds.) ICSOC/ServiceWave 2009. LNCS, vol. 6275, pp. 467–476. Springer, Heidelberg (2010)

15. Lane, S., Bucchiarone, A., Richardson, I.: SOAdapt: A Process Reference Model for Developing Adaptable Service-Based Applications. Information and Software Technology (2011)
16. Meiländer, D., Ploss, A., Glinka, F., Gorlatch, S.: A Dynamic Resource Management System for Real-Time Online Applications on Clouds. LNCS. Springer (2011) (to appear)
17. The Real-Time-Framework (RTF) (2011), http://www.real-time-framework.com
18. Glinka, F., Raed, A., Gorlatch, S., Ploss, A.: A Service-Oriented Interface for Highly Interactive Distributed Applications. In: Lin, H.-X., Alexander, M., Forsell, M., Knüpfer, A., Prodan, R., Sousa, L., Streit, A. (eds.) Euro-Par 2009. LNCS, vol. 6043, pp. 266–277. Springer, Heidelberg (2010)
19. Nurmi, D., Wolski, R., Grzegorczyk, C., et al.: The Eucalyptus Open-Source Cloud-Computing System. In: 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, pp. 124–131. IEEE Computer Society (2009)
20. Amazon Web Services (2011), http://aws.amazon.com

A Pragmatic Approach for Analysis and Design of Service Inventories

Patricia Lago and Maryam Razavian

Department of Computer Science, VU University Amsterdam, The Netherlands
{p.lago,m.razavian}@vu.nl

Abstract. Service Orientation is a paradigm for exposing business functions as reusable services and enabling business partners to discover those services on demand. Several service-oriented methodologies have been proposed in both academia and industry. Few methodologies, however, do provide concrete yet pragmatic support for developing software that is truly service-oriented, supporting service-specific aspects and challenges. As a consequence, the design of software services that are in-line with service aspects is left up to the prudence of the developers. To fill this gap, this paper proposes a methodology that incorporates essential ingredients of SOA during analysis and design. The methodology focuses service inventories while consciously addressing the perspectives of both service providers and service consumers. By supporting service-oriented reasoning, our methodology results in better reusable services, and the available inventories aid the development of service-oriented systems in a more cost-effective way. This is of the essence in cloud computing, which is a fast-spreading business model promising high ROI and cost savings.

1 Introduction

Nowadays, Service Orientation is having an impact on application development similar to that brought by Object Orientation in the nineties [1]. Both paradigms went through the *peak of inflated expectations* coined in the Gartner Technology Hype Cycle. While Object Orientation reached mainstream adoption long ago, Service Orientation is still, in our opinion, in the *slope of enlightenment* [2]: the *real* benefits of SOA are now becoming more widely understood, and more and more companies are migrating their software assets to SOA technologies.

In this maturation process, many so-called SOA methodologies have been defined in both industry and academia. Unfortunately, few (if any) do provide concrete support for the development of software that is *truly* service-oriented [3]. With term *service-oriented software* we mean both software services, and service-based applications (SBAs) reusing (i.e. composing) them. We argue that existing SOA methodologies have the same underlying assumptions as traditional software engineering methodologies, i.e. they assume that the developer both has ownership on the software (i.e. services), and has a system model in mind carrying holistic definition of all functionalities that should be produced. Both assumptions are of course not true in service-oriented (SO) development

G. Pallis et al. (Eds.): ICSOC 2011, LNCS 7221, pp. 44–53, 2012.

and cloud computing (CC) where services are neither owned nor always part of a "monolithic" system. According to [4], the introduction of the SO paradigm introduced a shift in the way we conceive a "software system": from a large system to a set of small pluggable elements. We therefore, argue that the subject of the architecting process should also shift from a system to the building blocks (i.e. the services); SO methodologies should be in-line with such shift. To support a software architecture following the service-oriented paradigm, a SO methodology should cover some essential ingredients:

Support Both Service- and Application Development. Developers of service-oriented software can play the role of either the service provider or the service consumer (or both). The *service provider* typically develops reusable software services. In doing that, it might reuse (or not) other available services, to build its own service compositions. In the latter case, the service provider switches to a service consumer role (described below) that develops more complex services. In any case, the output of the development is an *inventory of independent services* (or service pool) meant to be published for (internal or external) reuse. In this *service provider perspective* the development challenge is to identify *those* essential business functions that should be added to a service inventory, without knowing the business logic of the system/SBA that will reuse them.

The *service consumer*, instead, typically develops SBAs. In doing that, it always reuses services provided by third parties. The output of the development is in this case a software system meant to be purchased to customers and directly used by end-users. In this *service consumer perspective* the development challenge is to identify the characteristics of the services to be reused, and design and application that will dynamically discover and compose them, and still deliver a reliable overall system.

Focus on SOA. SOA is an architectural style that supports Service Orientation [4]. As such, it should (implicitly or explicitly) support (technical) mechanisms for service publication, dynamic discovery and composition.

Aim at Software Services. At a conceptual level, a *service* can be defined as a logical representation of a repeatable (business) activity that has a specified outcome (e.g. check customer data, provide weather report, deliver pizza, provide higher level education). A service is self-contained (state-less and adhering to a service contract); may be composed of other services (service composition); and is a black-box to its consumers.

Embrace the "Open World Assumption". Software services yield distributed ownership [5], i.e. they are often owned by (other) service providers. Developing (own) SBAs or service compositions hence implies that reuse is planned at design-time but can be actually tested only at run-time. Modern SO development must hence support a mix of design- and run-time development activities to make sure that dynamic aspects are engineered well and that the resulting software is reliable.

While all four ingredients described above are relevant, the major problem in current SOA methodologies is in the first one. The following describes our methodology for SO analysis and design of service inventories. The service

inventories, as such, provide the building blocks for a successful service offer following a SO paradigm. While we explicitly address the perspective of a service- or cloud provider developing inventories of software services (i.e. service provider perspective), we also need to support a service consumer perspective whenever services are to be reused from other providers. Both services and SBAs make up the service offer in a CC paradigm.

2 Terminology Overview

Services are the building blocks in our methodology. During the development lifecycle they get different forms. For the sake of clarity, in this section we define the different types of services as well as other basic concepts used in our methodology.

Conceptual Service: a conceptual service that is implementation agnostic. Considering the perspective of our methodology all services are conceptual as we do not address implementation of the services.

Business Service: a self-contained, stateless business function that accepts one or more requests and returns one or more responses through a well-defined, standard interface. During service identification, some elicited business services become the *service candidates* for design. They also might become *service operation candidates* to be clustered in service candidates.

Service Candidate: conceptual service identified during analysis that is a candidate software service.

Service Operation Candidate: a service operation identified from functional requirements; it might become a service candidate itself, or be composed (i.e. aggregated) in a (more complex) service candidate.

Task Service: service that mainly executes a functionality.

Entity Service: service that mainly manages, and offers access to, a (complex) data resource.

Hybrid Service: a mix of *task service* and *entity service*.

Utility Service: Domain- or application independent service offering access to generic functions or generic data resources. Also called *Infrastructure service*.

3 Methodology and Its Supporting Models

Service orientation is shifting the focus from engineering systems to integrating existing software services. During software development enterprises play both service provider and service consumer roles. Nevertheless, depending on the product they aim at producing (being a service inventory or a SBA), the methodology to be followed changes considerably. In this work we aim at producing service inventories. To this end, we identify the essential steps to carry out two phases: *SO analysis* is here defined as the process of determining how business requirements can be represented through business service candidates, while *SO design* is the process of modeling a software service inventory and/or reusing it to compose a SBA. In these phases, for modeling purposes we reused state-of-the-practice notations like UML and SoaML, and extended them only where they revealed to be

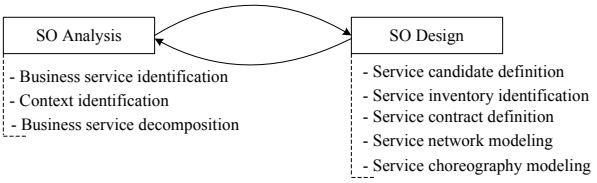


Fig. 1. Service Inventory Analysis and Design Methodology

insufficient. Fig. 1 depicts the steps of our SO Analysis and Design methodology, as further elaborated in the following.

Step 1: Business service identification. This step aims at identifying business services. These are elicited by use of business process models and conceptual data models. These two models can be determined either from functional models of pre-existing systems (i.e. bottom-up SOA migration), or from the target business domain, as the list of functional requirements (i.e. top-down service development).

Given the business processes, we identify business services by clustering service-relevant functionality within a business process. The elements of a business process model (e.g. activities and decision elements) are examined as candidate business services. The goal here is identifying the elements representing a self-contained functionality. Accordingly, business process models represent a sequence of business services, and hence, highlight potential orchestration behavior. To model business processes, we use UML activity diagrams (see Fig. 2(I))[1]. The clusters of functionality representing the business services are shown by dashed boxes. By modeling clusters in an explicit way, the analyst is helped in recognizing/eliciting relevant candidate services. Also, modeling clusters supports the analyst in comparing similar clusters for either merging them in a unified functionality, or keeping them separate.

The second model especially important during this step is the conceptual data model. This model facilitates the identification of the business services addressing the functionality of business data entities. The conceptual data model elements (i.e. data entities and relationships) are examined as candidate business services. These business services are highly reusable since they can be leveraged in independent business processes following the principle of separation of concerns and abstract data types (as in object orientation).

Step 2: Context identification. The main goal of this step is the identification of the set of external elements that the service inventory interacts with. Examples of such external elements are: external service providers, external services or SBAs, or external end users. In this step it is important to identify the external elements that rely on this business service inventory as well as those business services that rely on the external elements. The output of this step is a context model that illustrates the inventories in their environment. To

[1] In Fig. 2 we focus only on the models that are instrumental for the description of the methodology and that have been extended from standard UML and SoaML notations in order to support better analysis- and design reasoning.

draw context models we use the UML use case diagram notation. The context model represents the business services using *UML use cases*, the boundary of the inventory is illustrated using *use case system boundaries*, and the inter-relations among external entities and business services are shown by *associations*.

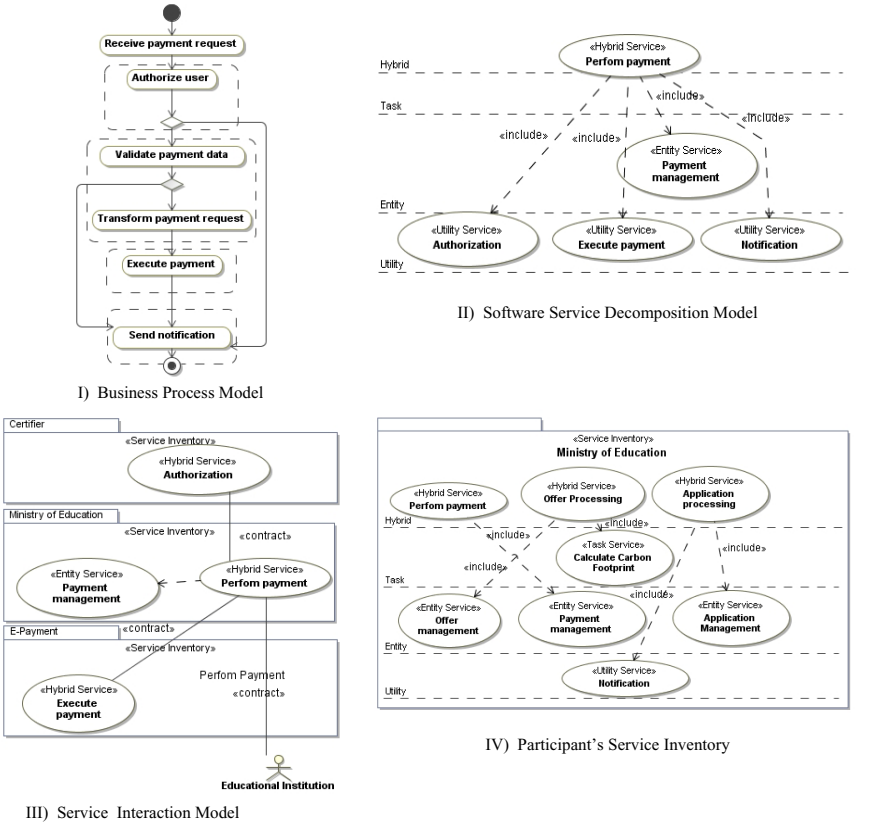


Fig. 2. Examples of Analysis and Design Models

Step 3: Business service decomposition. Given the set of identified business services, this step elicits the *candidate services* and their constituent *service operations*. This step results in the selection of the business services, identified during the previous step, that will be (partially) automatized by means of software services. Those business services are transformed into a number of candidate services or service operations. As the output of this step, each business service is modeled as a collection of candidate services. We model the decomposition of candidate services using the UML use case diagram notation.

Step 4: Service candidate definition. This is the first step of the SO Design phase. In this step, the service candidates of each business service (output of SO Analysis) are mapped on the service types (i.e. hybrid, task, entity and utility - refer to Section 2 for our definitions), from the perspective of a service provider.

As an example, Fig. 2.II shows the service candidates of the *Payment* business service with their corresponding service types: *Payment management* is an entity service managing the data about a payment, while *Authorization*, *Execute payment* and *Notification* are all utility services reusable across domains or applications. By identifying the service types the scope of reuse (i.e. domain-specific vs. domain-generic) of the services is identified. In this way, the service consumer is helped in reasoning about, e.g. if a service has been designed to be reused across different SBAs (utility service), or if it manages internally application-specific data (entity service) or application-specific activities (task service). This powerful yet simple modeling mechanism addresses an urgent problem in industrial practice, namely the need to reuse services developed for different purposes (e.g. in different projects or departments) and yet very similar in nature. Lack of support in this step leads to unnecessary duplication of services (silo problem) and governance problems, both major issues in IT service developing companies nowadays.

Step 5: Service inventory identification. During this step, decisions are made on which services are going to be provided by the inventories. Service inventory identification involves making a choice for the realization of service candidates. The realization alternatives include, but are not limited to, the following: *a)* reuse an existing element (e.g. component) and transform it as a service using wrappers;*b)* develop the candidate services from scratch;*c)* outsourcing the service realization; *d)* or purchasing, leasing, or paying per use (for) a service. The first three alternatives imply that the corresponding service will be provided by the service inventory. The last alternative, however, indicates that the service inventory, itself will be the consumer of that specific service which is provided by another provider. By mapping the candidate services on different service providers, the *participants* of the service candidates are identified. Fig. 2.IV represents the service inventory of the *Ministry of education* that consumes the services provided by *Certifier* and *e-Payment*. This way the participants of the Ministry of education service inventory are identified, i.e. *Certifier* and *e-Payment*. The output of this step is the service inventories representing the collection of software services of different types that each participant offers.

Step 6: Service contract definition. This step aims at identifying the service contracts between participants. As mentioned, cross-domain service interactions represent candidate service contracts. These interactions are inherently the ones crossing the participants' service inventories and the interactions can be between participants and external entities or between participants only. Given a specific business service, a contract represents how participants work together to realize the business service's goals. To identify the service contract this step models each business service with service interaction diagram (see Fig. 2.III). Using service interaction diagrams, the software services realizing each business service are mapped on their provider participant and further the interactions crossing participant's domains are identified as contracts, which are then modeled using 'SoaML contracts'. Thanks to this step, and as made explicit in the service interaction diagram, service providers are given with an holistic overview of

what their service inventory is offering to potential consumers and what are the existing (or to be established) service level agreements with third-party service providers. In this way, SO reuse and governance are supported. On the other hand, service consumers (or service providers planning to consume third-party services in the service compositions) are helped in reasoning about the impact of consuming a certain service offered by a (discovered) service provider.

Step 7: Service network modeling. Given the contracts identified in the previous step, this step models how participants work together to realize each business service using their contracts. The network architecture illustrates which participant is the provider of the contract and which one is the consumer. We model services networks using SoaML services network architecture diagram. It should be noted that, the services network architecture is modeled in a recursive manner. As such, a participant providing a service in a higher level network architecture can itself have a network architecture that shows how that service is provided using its software services.

Step 8: Service choreography modeling. Given the participants of each business service and service contracts among them this step defines the service choreography for each service contract. The service choreography model here represents the contract's behavior in terms of what is transmitted between the parties and when, without defining the internal behavior of the parties. Similar to service networks models, the choreography models are also defined in a recursive manner. This way, each participant defines the internal behavior of its own provided services. Service choreographies are modeled using SoaML choreography diagram and the internal behavior of the software services are modeled using the UML sequence diagram.

4 Extensions to UML/SoaML

UML/SoaML models revealed to be insufficient to properly support SO reasoning. To solve this problem, we extended some of the UML/SoaML diagrams by defining a UML profile, which implements the extensions to the diagrams used in the methodology. The following explains such insufficiencies and briefly describes the applied extensions.

Service Types. One of the shortcomings of SoaML is its lack of support of service types and their structuring. To fill this gap, our approach extends SoaML use case models in order to support mapping service candidates on service types. Each software service, modeled using use cases, is marked with a stereotype representing the type (see Fig. 2.II).

Service Inventory Model. Although service inventory is a first class element in SO design, SoaML does not support it. To fill this gap we use UML use case diagram and cluster software services to model the service inventories of each participant (see Fig. 2.IV) using packages marked with $<< ServiceInventory >>$ stereotype.

Service Interaction Model. While modeling collaborations through service contracts, SoaML also lacks in providing a model facilitating the identification of

service contracts to be designed. To fill this gap, we devised the service interaction diagram (see Fig. 2.III). As mentioned, cross-domain service interactions represent candidate service contracts. We model such interactions using use case associations marked with << *Contract* >> stereotype. It should be noted that in service interaction model, SBAs that just have the service consumer role can be directly modeled using UML actors.

Internal Service Behavior. Regarding service behavior, SoaML focuses on choreography only. Hence the internal behavior of the provided services is not covered. Considering the service provider perspective, both cross-domain behavior (modeled using SoaML contract-based choreography) and internal behavior of provided services need to be addressed. To cover the internal behavior, we exploit UML sequence diagram, which hence complete both perspectives of a SO design, that of a service provider developing its own services (or a service consumer developing its own SBAs) and that of a service consumer reusing externally provided services (or a service provider developing its own service compositions by reusing, again, externally provided (atomic) services).

5 Discussion

Many SO approaches for design and development of services have been proposed in both industry and academia. Some of these approaches support the entire service engineering process (e.g. [6,7]), while some others cover only a few phases such as service analysis and design (e.g. [8,9]). Our approach falls under the second category by specifically covering service analysis and design.

According to the survey carried out by Gu&Lago [3], the phases of service analysis and design are covered by most of the SOSE methodologies. Those methodologies, however, mainly provide guidelines rather than defining a fully fledged methodology that guides architects and designers step-by-step. For instance, most of the existing methodologies provide guidelines for service identification from different resources such as requirements or existing legacy systems. Those methodologies assume that by identifying the candidate services, the design of services can be done in a straightforward manner. Unfortunately, devising a solid design that supports requirements and goals and is in-line with SOA principles is not straightforward and needs to be aided in a step-by-step manner. SO methodologies should support the line of reasoning needed for achieving a service architecture design from the requirements. This is one main contribution of our methodology.

In our methodology we put special emphasis on modeling (in the most straightforward and pragmatic way possible) the elements that are left implicit in the existing SO notations and that in our experience aid reasoning. For example, modeling the clusters of activities in the Business Process Model of Fig. 2.I) helps analysing how to identify reusable business services; modeling the different inventories in the Service Interaction Model of Fig. 2.III) is necessary to elicit which dependencies among services cross different inventories, and therefore require the establishment of contracts; etc.. Thanks to such modeling extensions, we bridge the gap between the 'description of the artifacts' and 'how to create

those artifacts'. According to Tang [10], if a design problem is well-structured, designers tend to have a better reasoning. Our approach supports such structure in the inventory design problem, by supporting all essential seamless steps required for the reasoning throughout the service analysis and design. We have been using and refining it in the past five years, by teaching it to Master students and letting them apply it in their SO design practicals. As the students need to be trained in their reasoning, we observed the problems they were encountering and refined the way we support them in realizing their SO design. We also presented our methodology to various industrial partners who all expressed vivid interest to use it for their clients migrating to CC business models. We are currently applying it in a 700M Euro project in the field of airlines and airports, with great feedback so far.

Throughout the steps of our methodology, we support the perspectives of both the service provider and the service consumer. We support the service provider in seamlessly transforming business requirements into business services and all the way down to their supporting software services. In this way, we ensure that the resulting services expose the right functionality with the desired business value. In addition, we explicitly support the provider in analyzing its service inventory and the existing or needed contracts with third-party providers. We support the service consumer in identifying the characteristics of the services to be reused via e.g. service types, the explicitly clustering of business services, and their mapping on software services. Also, by supporting the identification and design of service contracts we support the provider that consumes external services for service compositions - this time the service provider and service consumer perspective together.

According to [3], most of the existing methodologies provide some guidance on the required models and notations. Those modeling methods, however, do not fully support the associated activities of the methodologies. This implies that some activities are not supported by the required modeling techniques. For instance, in order to identify the services contracts, one needs to identify the interactions among the services provided by the participants. To identify these interactions, a modeling technique must represent the participants, their provided services and the interactions among them. Although such a modeling technique is required, it is not supported by existing modeling notations. This reveals gaps in existing SO modeling notations. Our work identifies such gaps and fills these gaps by extending SoaML and UML.

Finally, by supporting the identification of business services and their transformation into supporting software services, our methodology helps bridging business requirements and IT solutions. This alignment of business and IT solutions has been one of the promises of SOA. Unfortunately, industry still suffers major problems in realising it. For example, while SOA promises to ease the communication of business- and IT stakeholders, business requirements coming top-down from the first are often not well understood by the latter, and hence realized ad hoc. This hinders effective reuse of software services and causes the 'silo problem' (e.g. similar business services across different development projects

are not identified and hence their development is duplicated). Ultimately, this causes governance problems. Our methodology is a first step to help stakeholders in achieving business-IT alignment by shaping the SO Design phase around the notion of business service.

6 Conclusion

In this paper we introduced a pragmatic modeling methodology focused on SO analysis and design. This methodology and its models stem in five years of teaching software- and service oriented design. It has been applied each year on different large industrial design projects and underwent continuous improvement. Recently we are using it also in collaboration projects with industrial partners, where it is catching increasing and positive attention.

Our methodology has various novelties. It is simple, pragmatic, focusing on the essential steps to go from business requirements to a design blueprint. It makes explicit the two service consumer- and provider perspectives by explicitly addressing the identification of service inventories and the offered services behavior (for providers), and by developing cross-domain interactions with services networks and contracts (for consumers). In this way, architects are compelled to *think and reason in a SO way* and are facilitated in better decision making driving migration toward CC business models.

Acknowledgments. This research received funding from projects Jacquard SAPIENSA (contract 638.001.206) and FP7 NoE S-Cube (contract 215483).

References

1. Cockburn, A.A.R.: The impact of object-orientation on application development. IBM Syst. J. 32, 420–444 (1993)
2. Gu, Q., Lago, P.: Exploring service-oriented system engineering challenges: a systematic literature review. Service Oriented Computing and Applications 3, 171–188 (2009)
3. Gu, Q., Lago, P.: Guiding the selection of service-oriented software engineering methodologies. Service Oriented Computing and Applications, 1–21 (2011)
4. Bell, M.: Service-Oriented Modeling: Service Analysis, Design, and Architecture. Wiley Publishing (2008)
5. Baresi, L., Di Nitto, E., Ghezzi, C.: Toward open-world software: Issue and challenges. Computer 39, 36–43 (2006)
6. Arsanjani, A., Ghosh, S., Allam, A., Abdollah, T., Gariapathy, S., Holley, K.: SOMA: a method for developing service-oriented solutions. IBM Syst. J. 47, 377–396 (2008)
7. Papazoglou, M.P., Heuvel, W.J.V.D.: Service Oriented design and development methodology. Int. J. Web Eng. Technol. 2, 412–442 (2006)
8. Zimmermann, O., Krogdahl, P., Gee, C.: Elements of Service-Oriented Analysis and Design (2004)
9. Allen, P.: SOA best practice report: the service oriented process. Technical report, CBDi Journal (2007)
10. Tang, A.: Software designers, are you biased? In: Proceedings International Workshop on Sharing and Reusing Architectural Knowledge, p. 8. ACM (2011)

Artifact-Centric Modeling Using BPMN

Niels Lohmann and Martin Nyolt

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

Abstract. BPMN offers a rich pool of language constructs to model different aspects of choreographies, interorganizational business processes and service compositions. With collaborations and choreographies, BPMN enables the modeler concentrate on the control flow and the message flow, respectively. At the same time, data flow is only treated as a subordinate extension. In contrast, recent *artifact-centric* approaches model processes from the point of view of the data objects that are manipulated during the process. This paper investigates to what extend BPMN is suitable to model artifact-centric processes and which extensions are required to comfortably support this modeling approach.

1 Introduction

Business process modeling includes a specification of the order in which tasks are executed (control flow), the way data are processed (data flow), and how different branches in distributed and interorganizational business processes and services are invoked and coordinated (message flow). The current BPMN standard offers two different views on business processes: (1) collaboration diagrams (sometimes called interconnected models) that emphasize the local control flow of each participant of the process and (2) choreography diagrams (interaction models) that describe the process from the point of view of the messages that are exchanged among the participants. Conceptually, collaboration diagrams can be seen as control-flow centric models whereas choreography diagrams follow a message-flow centric view. In either diagram, data flow — if at all — only plays a subordinate role.

As third school of thought, artifact-centric models do not specify processes as a sequence of tasks to be executed or messages to be exchanged (i. e., imperatively), but from the point of view of the data objects (called *artifacts*) that are manipulated throughout the course of the process (i. e., declaratively). In recent work [10], we showed that artifact-centric models can be automatically transformed into choreographies and collaborations while guaranteeing certain correctness criteria such as soundness or compliance to business rules [9].

There currently does not exist a common conceptual modeling language for artifact-centric processes. In this paper, we investigate whether BPMN is suitable to express data aspects and to which extend BPMN needs to be extended to be used in an artifact-centric setting. The choice to study BPMN is motivated by its flexibility, comprehensibility, and its popularity among domain experts.

G. Pallis et al. (Eds.): ICSOC 2011, LNCS 7221, pp. 54–65, 2012.

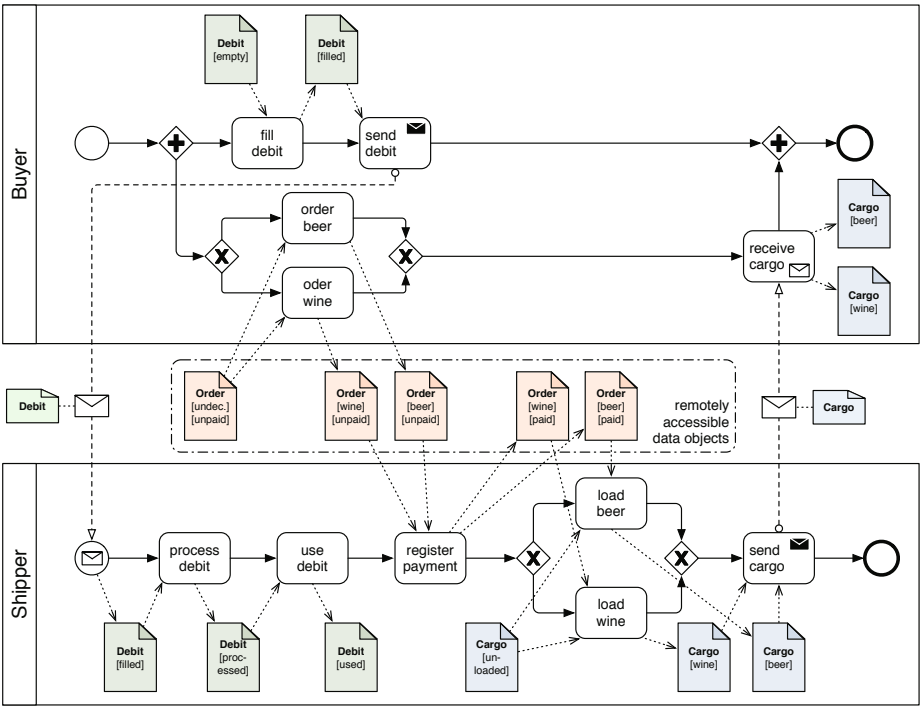


Fig. 1. Shipping scenario as BPMN collaborative process

Organization. The rest of this paper is organized as follows. The next section introduces a shipping scenario and discusses several issues in BPMN's capability of modeling data flow and manipulation. In Sect. 3, we briefly introduce artifact-centric modeling. The main contribution of the paper is presented in Sect. 4. We introduce several BPMN extensions to model the different aspects of artifact-centric processes. We continue by discussing the resulting model and related work in Sect. 5 and 6, before Sect. 7 concludes the paper.

2 Data Flow Modeling in BPMN

We shall employ a shipping scenario taken from [10] as running example for this paper. It consists of a buyer that (1) fills a debit order and (2) may choose between ordering wine or beer and a shipper who (3) processes the debit order, (4) loads a cargo with the ordered goods, (5) and sends it to the buyer. Figure 1 depicts a collaborative BPMN process model of this scenario.

Several data objects, or *artifacts*, are involved in the process: the debit, the order, and the cargo. Apparently, the cargo is not a classical data object, but a physical artifact. Nonetheless, a conceptual model should be able to cope with such objects. *This paper does not focus on actual implementations which would*

be done in standard languages such as WS-BPEL. The state of these artifacts can be changed by tasks. For instance, the buyer's task "fill debit" changes the debit's state from "[empty]" to "[filled]". Usually, the access to artifacts is restricted and an artifact is implicitly owned by a participant of the process. At the same time, it may be necessary to change the location of an artifact; that is, to send it to another participant. In the example, the debit is eventually sent to the shipper. As BPMN does not support this directly, we used an association to specify that the debit artifact *is* actually the message that is sent to the shipper. Beside artifacts that may change their location, it is also common to assume artifacts that can be remotely accessed. Such artifacts do not have a canonic owner or pool to be associated to. In the shipping scenario, the order artifact is an electronic document that can be remotely accessed by both participants: It does not need to be explicitly sent to the shipper to be evaluated. Again, BPMN does not define a canonic way to specify this. Finally, artifacts may contain several data fields that can be manipulated independently. The order artifact in the example process contains information on the desired goods ("unspecified", "wine", or "beer") and the payment status ("unpaid" and "paid"). A partial state change (e. g., setting the payment status to "paid" without considering the ordered items) is not supported by BPMN, which makes the shipper's "register payment" task appear clumsy and underspecified.

The shipping scenario shows that BPMN's capability to specify data objects and data flow is rather limited and a shift to an artifact-centric approach is impossible without making several extensions and adjustments.

3 Artifact-Centric Modeling in a Nutshell

Artifact-centric modeling promotes the data objects of a process and their life cycles to first class citizens. An artifact-centric model of the shipping scenario is consequently specified from the point of view of the *artifacts*; that is, the order, the debit, and the cargo. As we see in Fig. 1, the states of the artifacts may evolve over time, and each state change is performed by an *agent*, namely the buyer or the shipper. Specifying which agent may change an artifact's state also requires information on the location and the access control of an artifact: Physical artifacts (such as the cargo) need to be sent to an agent to be processed, whereas logical artifacts (such as a data base) can be remotely accessed.

The life cycle of each artifact can be seen as a small business process with an initial state and at least one final state. The latter models a successful completion such as "order paid" or "cargo loaded". As the artifacts can evolve independently, the control flow of the whole business process is specified *declaratively.* Each execution that brings each artifact to a final state can be seen as sound. Apparently, this includes unreasonable executions, for instance those where an implicit order of tasks is violated (sending the cargo before placing an order). Also the final states of artifacts may not be arbitrarily mixed, because otherwise executions in which beer is ordered and wine is delivered would be possible. In recent work [10], we proposed *policies* and *goal states* to exclude such undesired

behavior: Policies constrain the execution order of tasks in different artifacts and goal states exclude certain combinations of artifacts' final states.

From the artifact life cycles, the policies, and the goal states, a process model (such as the one in Fig. 1) can be *automatically synthesized* which is sound by design; that is, correctness of the model follows from the synthesis algorithm. The interested reader is referred to [10] for a detailed discussion.

4 BPMN Extensions for Artifact-Centric Modeling

In this section, we give more details on the ingredients of artifact-centric processes and propose BPMN extensions for each aspect. In particular, we consider

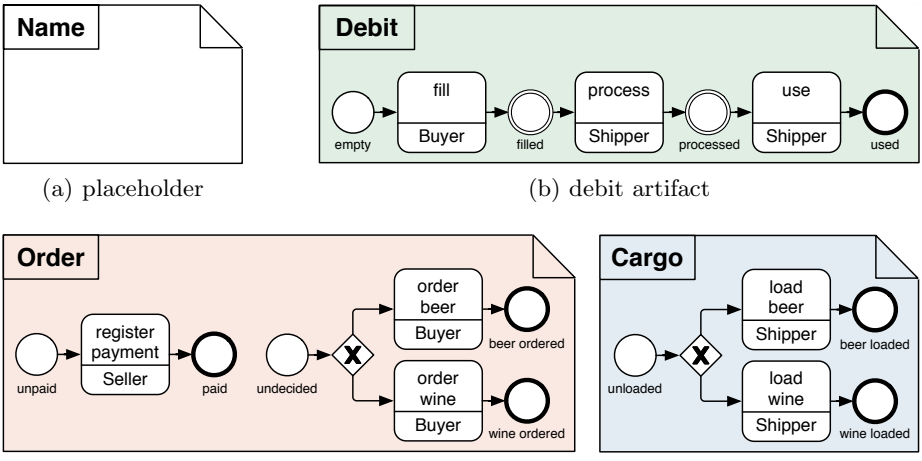
- artifacts: the process model's basic building blocks;
- object life cycles: a specification of the artifacts' states;
- location information: a means to specify how artifacts change their location;
- access control: a specification of remote accessibility of artifacts;
- goal states: a specification of desired final states; and
- policies: a means to remove undesired behavior.

The structure of this section can be seen as a recipe specifying the natural order in which artifact-centric models are created. We thereby exploit that the artifact-centric approach is modular in the sense that artifacts can be specified independently from one another. Hence, each artifact can be refined locally, and later synchronization (e. g., removing undesired behavior by applying policies) usually only affects a small subset of all artifacts.

4.1 Artifacts and Object Life Cycles

The basic building blocks of the artifact-centric approach are *artifacts* which are represented just like data objects in BPMN, cf. Fig. 2(a). The artifact's name is noted in the upper left corner. Without any further information, it can be treated as a placeholder symbol just like an empty pool in a collaboration. This placeholder symbol can also be used to hide details, for instance when policies are modeled and the focus lies on the dependencies among artifacts.

The object life cycle of an artifact (cf. Fig. 2(b)–2(d)) can be modeled using the symbols for tasks, events, and gateways. However, artifacts have no behavior per se, but are passive objects whose state changes are triggered by agents, for instance "Buyer" or "Shipper". Consequently, each task is annotated by an agent who may execute it. Furthermore, events are annotated by adjectives that describe the current state of the artifact (e. g., "empty" or "filled"). Initial and final states are denoted using the standard BPMN symbols for start and end events, respectively. Note that the life cycle of an artifact may have more than one symbol denoting an initial state. As an example for this, consider for instance the order artifact in Fig. 2(c): This artifact keeps track of the payment status and the purchase order. Each of these information has an individual initial state. The order is initially "unpaid" and the purchase order is initially "undecided".



(a) placeholder (b) debit artifact

(c) order artifact (d) cargo artifact

Fig. 2. BPMN representation of artifacts and object life cycles

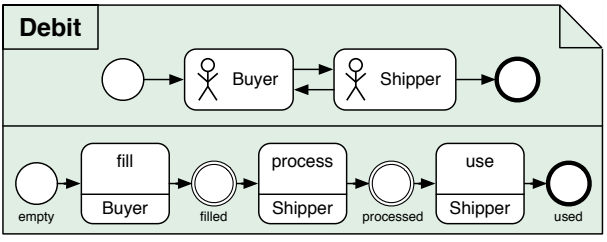
As we can see from Fig. 2, the local life cycles of the artifacts are rather brief and clear compared to the data handling in the process of Fig. 1. We achieved this by "abusing" control flow constructs to model the evolution of object life cycles. As the semantics of the gateways remains the same, the object life cycle models should be intuitively understandable by BPMN modelers.

4.2 Location Information

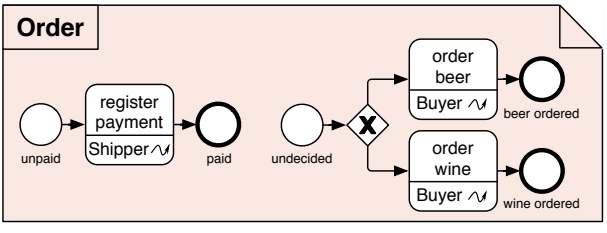
We already sketched in Sect. 3 that artifacts may be logical or physical objects that need to be transported between agents to perform state changes (e. g., the cargo artifact) or data objects that can be remotely accessed (e. g., the order artifact). This differentiation has an impact on the execution of the process, because artifacts may need to be sent to an agent before he can execute a task.

To this end, we extend the artifact models with *location information* and information about accessibility. We assumed that the debit artifact can be sent among the buyer and the shipper. This is modeled by an additional life cycle in the upper part of the debit artifact, cf. Fig. 3(a). Thereby, each agent is treated as a location (depicted by a stick-figure) and the arrows specify message channels between these locations. In addition, a start event and an end event specify that the debit artifact resides initially at the buyer and a successful processing of the artifact is only possible at the shipper agent. The cargo artifact is extended similarly. When synthesizing a process such as the model in Fig. 1, respective message events are inserted automatically and make sure that, for instance, the debit artifact is sent to the shipper before the task "process debit" is executed. Again, we refer to [10] for more information on the synthesis.

In contrast, the order artifact does not have a location, but can be remotely accessed, for instance using an URL or a different form of addressing mechanism.



(a) debit artifact with location information (message exchange)



(b) order artifact with location information (remote access)

Fig. 3. BPMN representation of location information

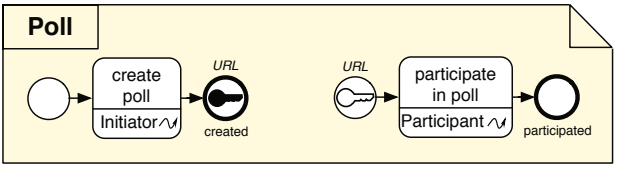


Fig. 4. BPMN representation of the creation and distribution of access control tokens

This is depicted by a small lightning symbol next to the agent's names, cf. Fig. 3(b). This means that the agent can always execute the task and does not need to receive the artifact before.

4.3 Access Control

Up to now, we assumed that an artifact is either an object — physical or logical — that needs to change its location to be accessed or a remotely accessible data object whose location (e. g., its URL) is known to the agents. The latter abstraction fails short in describing situations in which data objects are created or the access is only granted after other tasks have been executed. In such situations, the *access control* must be explicitly modeled.

As an example, consider the Doodle Web service (`http://doodle.com`) to schedule meetings or events. After creating a poll, the service returns a URL to that poll that can be sent to colleagues that should take part in the meeting. Without this URL, a participation in the poll is impossible, and the URL can hence be seen as an access token. We visualize this access control granting by a

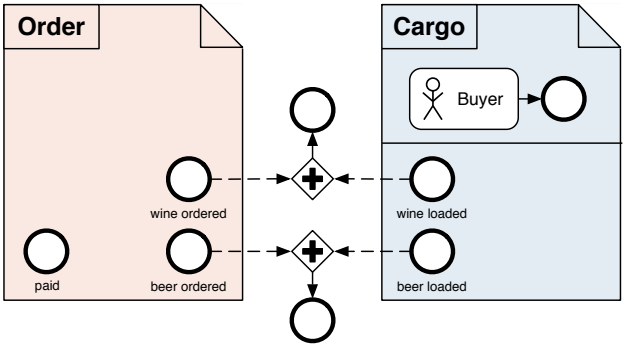


Fig. 5. BPMN representation of goal states

novel event type with a key symbol, cf. Fig. 4. In the example, the initiator of the poll creates a key with name "URL", and the participant can only execute the task "participate in poll" after receiving this URL.

The extensions we presented so far only affect single artifacts. Hence, each artifact can be modeled from a mere placeholder to a fully specified model including object life cycles, location information, and access control. The remaining extensions concentrate on the interdependencies between artifacts.

4.4 Goal States

Up to now, any execution of tasks that lead each artifact to a final state would be seen as a successful outcome of the shipping process. Such executions would, however, include runs that reach the final state "beer ordered" of the order artifact (cf. Fig. 2(c)) and "wine loaded" of the cargo artifact (cf. Fig. 2(d)). Apparently, such executions are undesired as the final states of the artifacts do not match. To exclude such undesired combinations, we specify *goal states*. A goal state is a combination of artifact's final states, and each combination that is not mentioned as goal state is implicitly excluded during the synthesis to a collaborative process model.

To give goal states a BPMN representation, we connect the desired end events of the artifacts with a parallel gateway, cf. Fig. 5. In this example, connecting "beer ordered" and "beer loaded" respectively "wine ordered" and "wine loaded" has the effect that only these combinations are valid. Considering all artifacts, this yields two valid final states of the overall process:

1. The order is paid and beer was ordered. The cargo is at the buyer and loaded with beer. The debit is used and at the shipper.
2. The order is paid and wine was ordered. The cargo is at the buyer and loaded with wine. The debit is used and at the shipper.

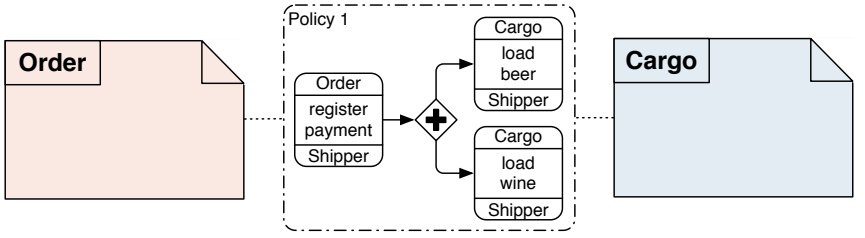


Fig. 6. BPMN representation of policies

In Fig. 5, we used placeholder symbols for the order and the cargo artifacts and only depicted the end events. Such a view on the end events should be provided by a modeling tool to make the specification of goal states as simple as possible.

4.5 Policies

Artifact-centric approaches follow a declarative modeling style: Instead of explicitly modeling global state changes, we only modeled the local object life cycle of each artifact. Consequently, the order of tasks in the generated process is only constrained with respect to goal states. As a downside of this approach, a lot of unreasonable behavior is exposed. For instance, sending out an unloaded cargo or even sending without prior payment is possible. To rule out this undesired behavior, we employ four *policies*:

1. Only load the cargo after payment has been registered.
2. Only register the payment when the filled debit form is at the shipper.
3. Do not send an unloaded cargo to the buyer.
4. Only send the debit form if it is filled and at the buyer.

A policy specifies dependencies between the tasks of one or more artifacts. For instance, the first policy expresses a causality between the order and the cargo artifact. To model policies in BPMN, we use tasks and gateways to express the additional dependencies and add an association to all involved artifacts. As an example, consider Fig. 6. In the upper part of each task, we also specify the artifact it belongs to. Policy 1 specifies that the "register payment" task of the order artifact must be executed before the "load beer" or the "load wine" task of the cargo artifact are executed. The parallel gateway is used, because a policy should not exhibit choices (i. e., we do not allow exclusive splits). The fact that the last two tasks are mutually exclusive follows from the life cycle of the order artifact, cf. Fig. 2(c). Again, we used placeholder symbols for the involved artifacts and a modeling tool should support the hiding of the object life cycles. The other policies can be modeled similarly.

5 Discussion

Figure 7 depicts the overall artifact-centric model. It consists of the three artifacts (with their object life cycles and the location information), four policies, and

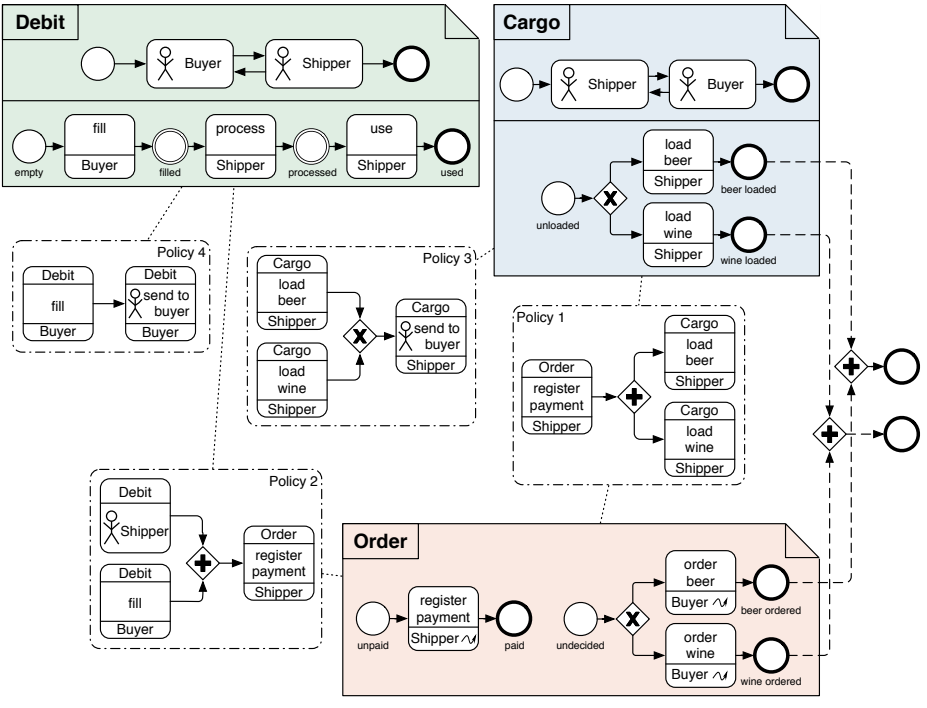


Fig. 7. Complete artifact-centric BPMN model of the shipping scenario

two goal states. This model can be automatically transformed into the process depicted in Fig. 1 using the synthesis approach described in [10]. Conceptually, this synthesis is defined in terms of Petri nets [12], but the BPMN constructs can be straightforwardly mapped to Petri nets using the formal semantics of Dijkman et al. [6]. More details can be found in [11].

At this point, a comparison between the artifact-centric model (cf. Fig. 7) and the collaborative process model (cf. Fig. 1) is difficult and premature, because without empirical studies, it is impossible to decide which model is easier to understand or which model can be created in a shorter period of time. In particular, such questions heavily rely on the education of the modelers and proper tool support. However, we would like to mention some aspects that are unique to, or at least enforced by, artifact-centric process models.

Modularity. We already mentioned that artifacts can be modeled independently. That is, a modeler only needs to concentrate on a single artifact model at a time. This partition into smaller, "brain-sized", units may not only improve the understandability and the model quality, but should also facilitate the concurrent modeling of all required artifacts of a process. Furthermore, adding or refining an artifact does not influence the other artifacts which adds flexibility to the approach.

The moment dependencies between several artifacts need to be expressed, views may help the modeler focus on the relevant aspects. For goal states, only the artifact's end events need to be considered. If a tool supports such a view, the specification of a goal state should boil down to a few selection actions. Likewise, the creation of a policy only depends on the names of the artifact's tasks and could be facilitated by a modeling tool by corresponding menus and support for autocompletion.

To summarize, the modularity allows to explore and create a model gradually. Compared to the model in Fig. 1, less global dependencies need to be understood at once. As a side effect, the artifact-centric model (cf. Fig. 7) is already graphically partitioned into smaller units.

Declarative modeling. The declarative modeling style is a liberal specification of all possible behavior. Thereafter, undesired behavior needs to be ruled out by adding goal states and policies. Given the artifacts, the goal states, and the policies, a sound process model can be automatically synthesized. In case an aspect changes, this synthesis can be repeated. This may increase the flexibility and the modeling speed of this approach. Even if the change *affects* large parts of the process, only a few parts of the original artifact-centric model need to be changed due to the modularity. For instance, adding the possibility to perform a third-party debit check would have a large impact on the overall process, but could be realized by a small adjustment to the debit artifact.

The automatic synthesis further allows to abstract from certain error-prone aspects such as message protocols: The message flow directly follows from the location information. It ensures that artifacts are only sent to agents to perform tasks on them or to satisfy goal states.

6 Related Work

Since the first draft of BPMN, several extensions have been proposed that aim at giving modelers the constructs at hand to model additional aspects of their processes: Decker et al. [5,3] studied extensions toward choreography modeling. These extension have already been picked up by the OMG and choreography models are now a part of the current BPMN standard. Other extensions focus on nonfunctional properties such as performance measures [8], time requirements and constraints [7], service quality requirements [14], authorization [15] and security [13], transactions and compensation [2], or process compliance [1].

All extensions share the evaluation that BPMN is the de facto standard in process modeling and that it is promising to integrate novel aspects to BPMN rather than to propose an alternative language. This results in careful extensions that try to reuse BPMN constructs as much as possible and to make extensions appear as natural as possible. Best practices such as abstraction by sub processes or hiding of unnecessary details are also frequently adopted. Depending on whether the extension is just conceptual or already aimed at execution, also the meta model needs to be adjusted.

To the best of our knowledge, this paper provides the first extension of BPMN toward artifact-centric modeling. As our extensions are at a conceptual model, we did not specify an extension to the BPMN meta model at this point.

7 Conclusion

Summary. We investigated a small process model of a shipping scenario and showed that certain data-flow aspects are not faithfully supported by the current BPMN standard. To model artifact-centric processes, we proposed several BPMN extensions to represent the parts of the artifact-centric approach. Thereby, we tried to reuse existing graphical notations as much as possible. We also tried to follow modeling best practices and define different views on the process to help the modeler focus on relevant details.

Lessons learnt. The current BPMN standard has only limited modeling support for data aspects. In particular, modeling the life cycle of data objects results in cluttered models. By "abusing" control flow constructs such as tasks, events, and gateways, artifacts and their life cycle can be effectively modeled. As a matter of fact, the whole school of artifact-centric process design can be expressed with only a few adjustments to BPMN.

As artifact-centric models are naturally partitioned into smaller parts (i. e., artifacts, policies, and goal states), they facilitate a gradual modeling approach. Beside the modularity, the declarative nature further improves flexibility, because even if changes affect large parts of the model (e. g., if a policy or an artifact changes), a sound process can be automatically synthesized. This process can then be realized by several services; that is, the process is replaced by a service composition.

Future work. This paper is only the first step toward BPMN support for artifact-centric modeling. One obvious direction of future work is the integration of the proposed extensions into a BPMN modeling tool such as Oryx [4]. Beside the mere support of the concepts, also an implementation of different process views (i. e., a view on a single artifact, a view on goal states, and a view to model policies) is required.

Based on a prototyping implementation, case studies and empirical experiments need to be conducted. The question which kind of processes may benefit from an artifact-centric modeling approach is still open. This includes questions regarding modeling speed, model quality, adaptability and flexibility, as well as understandability of the models.

References

1. Awad, A., Decker, G., Weske, M.: Efficient Compliance Checking Using BPMN-Q and Temporal Logic. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 326–341. Springer, Heidelberg (2008)

2. Bocchi, L., Guanciale, R., Strollo, D., Tuosto, E.: BPMN Modelling of Services with Dynamically Reconfigurable Transactions. In: Maglio, P.P., Weske, M., Yang, J., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6470, pp. 396–410. Springer, Heidelberg (2010)
3. Decker, G., Barros, A.: Interaction Modeling Using BPMN. In: ter Hofstede, A.H.M., Benatallah, B., Paik, H.-Y. (eds.) BPM Workshops 2007. LNCS, vol. 4928, pp. 208–219. Springer, Heidelberg (2008)
4. Decker, G., Overdick, H., Weske, M.: Oryx – An Open Modeling Platform for the BPM Community. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 382–385. Springer, Heidelberg (2008)
5. Decker, G., Puhlmann, F.: Extending BPMN for Modeling Complex Choreographies. In: Meersman, R., Tari, Z. (eds.) OTM 2007, Part I. LNCS, vol. 4803, pp. 24–40. Springer, Heidelberg (2007)
6. Dijkman, R.M., Dumas, M., Ouyang, C.: Semantics and analysis of business process models in BPMN. Information & Software Technology 50(12), 1281–1294 (2008)
7. Gagné, D., Trudel, A.: Time-BPMN. In: CEC 2009, pp. 361–367. IEEE (2009)
8. Korherr, B., List, B.: Extending the EPC and the BPMN with business process goals and performance measures. In: ICEIS 2007, pp. 287–294 (2007)
9. Lohmann, N.: Compliance by Design for Artifact-Centric Business Processes. In: Rinderle-Ma, S., Toumani, F., Wolf, K. (eds.) BPM 2011. LNCS, vol. 6896, pp. 99–115. Springer, Heidelberg (2011)
10. Lohmann, N., Wolf, K.: Artifact-Centric Choreographies. In: Maglio, P.P., Weske, M., Yang, J., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6470, pp. 32–46. Springer, Heidelberg (2010)
11. Nyolt, M.: Modellierung artefaktzentrierter Geschäftsprozesse. Bachelorarbeit, Universität Rostock, Rostock, Germany (2011), (in German)
12. Reisig, W.: Petri Nets. EATCS Monographs on Theoretical Computer Science edn. Springer (1985)
13. Rodríguez, A., Fernández-Medina, E., Piattini, M.: A BPMN extension for the modeling of security requirements in business processes. IEICE Transactions 90-D(4), 745–752 (2007)
14. Saeedi, K., Zhao, L., Sampaio, P.R.F.: Extending BPMN for supporting customer-facing service quality requirements. In: ICWS 2010, pp. 616–623. IEEE (2010)
15. Wolter, C., Schaad, A.: Modeling of Task-Based Authorization Constraints in BPMN. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 64–79. Springer, Heidelberg (2007)

Migratability of BPMN 2.0 Process Instances*

Angineh Barkhordarian, Frederik Demuth, Kristof Hamann,
Minh Hoang, Sonja Weichler, and Sonja Zaplata

Distributed Systems and Information Systems, Department of Informatics
University of Hamburg, Germany
hamann@informatik.uni-hamburg.de

Abstract. The migration of running process instances allows for a dynamic distribution of individual business processes at runtime. However, a widely-used standardized process description language and an agreed format for the exchange of process instance data are vital for the applicability of such concept. The newly evolved standard of the *Business Process Model and Notation (BPMN 2.0)* is currently gaining acceptance in many organizations and is supported by a growing number of process engines. In order to leverage BPMN for the dynamic distribution of business processes, this paper presents an analysis on the migratability of running BPMN process instances. The results include a mapping of BPMN 2.0 control flow elements to an existing migration model and a novel migration concept for process instances which contain BPMN-specific elements such as events, pools and user tasks. In addition, the effort for extending a BPMN process engine is evaluated by a prototype implementation based on the open source *Activiti process engine*.

1 Motivation

In today's dynamic environments, business processes are often subject to changes related to the content and the structure of the business case. In consequence, flexibility of supporting information and communication systems is one of the most driving factors. Considering that a business process is not always executed by a single organization or, more technically, by a single process engine, also the requirements for the distributed execution of individual process instances can change dynamically. A typical example is the spontaneous shift of a selected process partition to a mobile device in order to allow for its offline execution in a different location [3,13]. Other examples include the dynamic distribution of process instances due to load balancing strategies for process engines [12] or the runtime exchange of business partners in order to quickly react to market changes or to individual demands of customers [11].

In situations where requirements for a distributed execution of individual process instances can change dynamically, a flexible distribution mechanism is needed. In comparison to other distribution models which require to determine

* The research leading to these results has received funding from the European Community's Seventh Framework Programme under grant agreement 215483 (S-Cube).

G. Pallis et al. (Eds.): ICSOC 2011, LNCS 7221, pp. 66–75, 2012.

process partitions at design time (e.g. web service choreographies), the concept of *process instance migration* allows for deciding about most distribution parameters at the runtime of each process instance [6,10,14]. The procedure used here basically involves stopping the execution of a running process instance, capturing its current status, and transferring both the process model and its instance data to another process engine where the execution of the process is continued. Using this strategy, a business process can be distributed at nearly any time with an arbitrary granularity of process partitions and open number of potentially participating process engines [14].

As such dynamic distribution requires a common understanding of the business case as well as a standardized specification including an execution semantic, the applicability of process instance migration is dependent on the underlying process description language, its specific structure and its control flow elements. A promising candidate is the newly evolved standard of the *Business Process Model and Notation (BPMN 2.0)* [9] which is – in contrast to its predecessor BPMN 1.1 – not limited to a graphical representation of business process models but also provides an execution semantic. It therefore closes the gap between business process analysts and technical developers and thus significantly reduces the time between the development phases of design and implementation. Although BPMN also integrates elements to describe distribution (i.e. *pools* and *lanes*), a runtime migration of BPMN process instances has not yet been considered. Instead, the distribution of BPMN processes is still an inflexible and time-consuming procedure because it requires the manual deployment of predetermined process partitions on pre-selected process engines prior to runtime.

In order to address these issues, this paper presents an analysis on the general *migratability* (i.e. the ability to partition and transfer the process within a specific control flow structure) of running BPMN process instances based on an existing generic migration model which allows the representation of language-independent process instance data. Relevant background information about the migration model and related approaches are presented in Section 2. Section 3 discusses the specific characteristics of BPMN language elements and proposes a mapping of these elements to the migration model. Section 4 evaluates the effort for enhancing an existing BPMN process engine in order to realize the migration within a prototype implementation. Results are summarized in Section 5.

2 Background and Related Work

Process instance migration as a basic concept for distributed workflow management (not to be confused with migration of instances to another schema, e.g. *case migration* [5]) has been introduced by Cichocki and Rusinkiewicz [6] in 1997. More recently, the framework *OSIRIS* [10] relies on passing control flow between distributed workflow engines in order to execute service compositions. In *Adept Distribution* [4] a similar approach to process fragmentation is presented which supports dynamic assignment of process parts to so-called *execution servers*. The control of a particular process instance migrates from one execution server

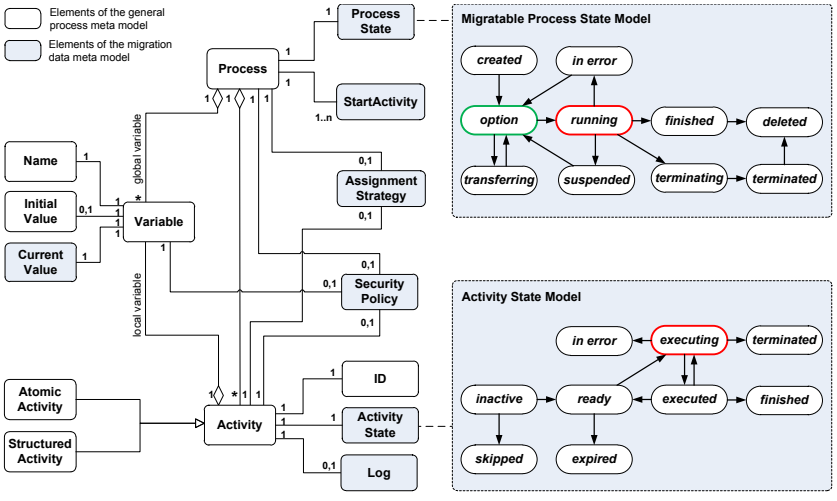


Fig. 1. Generic migration model (simplified) [14]

to another and process activities can influence the next participant. Related to this, Atluri et al. [1] present a process partitioning algorithm which creates self-describing subprocesses allowing dynamic routing and assignment. Process migration has also particularly been applied to the area of mobile process execution, e.g. by Montagut and Molva [8]. Their approach relies on passing control flow between distributed WS-BPEL engines in order to access applications internal to mobile devices. Similarly, the *DEMAC* middleware [13] is able to delegate process execution (in whole or in part) to other stationary or mobile process engines. However, the presented approaches either propose a *migration-specific extension* of a (standard) process description language such as XPDL or WS-BPEL (e.g. [8,13]) or use a *completely proprietary specification* (e.g. [6,10,4,1]). Based on its novelty as an executable process description language, migration of (unmodified) BPMN process instances has not been considered yet.

As a more general approach, a technology-independent migration model has been introduced in [14] (cp. Figure 1). Here, the migration model only assumes the existence of a common minimal process model consisting of a finite number of *activities* representing the tasks to be fulfilled during process execution, and a finite number of global or local *variables* holding the data used by these activities. Activities are either *atomic activities* (i.e. represent a specific task) or *structured activities* (i.e. a control flow structure as a container for other activities). A process description complying to these properties can be supplemented with a separate description of migration data documenting the state of the variables (*current value*) as well as the execution state of the process (*process state*) and of each activity (*activity state*). In order to preserve the process's consistency and the integrity of its data, a process instance is not allowed to be migrated until all of the currently executed atomic activities are completed. Thus, the process lifecycle state *option* defines a stable point to transfer a process during its

execution which can only be reached if none of the activities are in the activity lifecycle state *executing*. In order to control the selection of target execution systems, each activity of the process can be optionally connected to an *assignment strategy* and/or a *security policy* representing user defined distribution strategies and restrictions. In addition, a set of activities can be referenced as *start activities* to mark the first activities to be executed after process migration. In order to connect the migration data model with the original process model, each activity and variable must have a unique name or identifier (ID). As BPMN complies to these basic requirements, this generic migration model can be used as a basis to analyse the migratability of BPMN process instances.

The concept of process instance migration has recently been evaluated for basic elements of XPDL and WS-BPEL processes (cp. [14]). However, regarding the ability for migration, BPMN processes have inherently different characteristics and thus impose new challenges for migration. In contrast to WS-BPEL, BPMN processes are able to combine interactive user-centric tasks and automatic application tasks such as web services on both a specific or an abstract level. Furthermore, BPMN processes can already specify descriptions about a distributed execution and BPMN includes a strong event concept allowing event-based control flow constructs. In order to address these differences in more detail, the following section discusses relevant BPMN control flow elements with respect to a potential migration of the process.

3 Migrating BPMN Processes

The *Business Process Model and Notation* (*BPMN*) is a graph-based description language for the specification of business process models which can be expressed in a standardized graphical notation, and, since BPMN 2.0 [9], also in a respective executable XML representation. Its goal is to provide a single specification for notation, metamodel and interchange format of business processes. Following BPMN, a process consists of a number of flow objects (*tasks*, *gateways* and *events*), connecting objects (*sequence flows* and *message flows*), distribution objects (*pools* and *swimlanes*) and other artifacts (e.g. *data objects* and *groups*) (cp. Figure 2 for an overview). The migratability of the most important control flow constructs is discussed in the following.

3.1 Tasks

A BPMN process consists of at least one functional *task* which represents a single unit of work to be done. Considering the migration model presented in Section 2, a *task* corresponds to an *atomic activity*. However, in contrast to e.g. WS-BPEL activities, BPMN tasks can be defined on different levels of abstraction and do not require to bind a resource (e.g. a web service or a human process participant) by a static binding. Instead, resources can also be specified by a role or a number of characteristics and thus can be rebound after migration. Depending on the assignment strategy, a task can be rebound to a locally-available resource at the target system, the local binding can be replaced by a unique systemwide reference

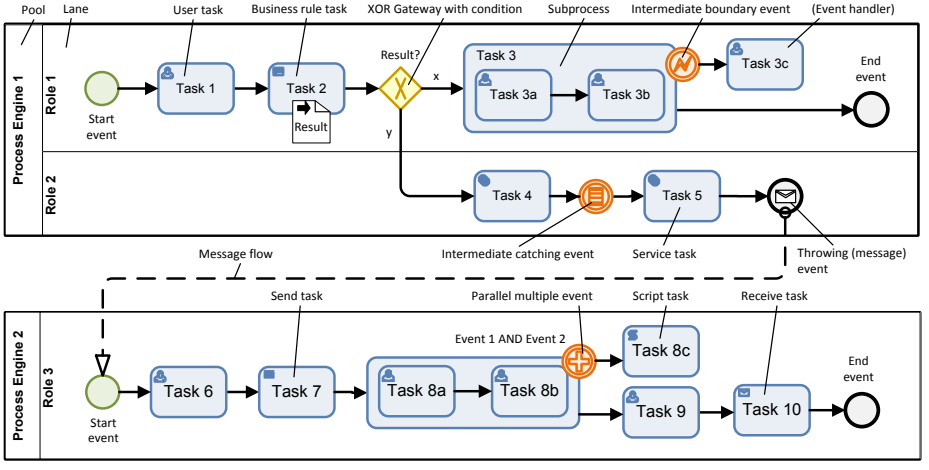


Fig. 2. Example of BPMN elements and control flow constructs

or, if applicable, the resource can be moved or copied to the target system [7]. In consequence, this abstraction implies more possibilities for migration and thus possibly leads to a higher level of flexibility.

In more detail, BPMN supports different types of tasks. *Service tasks* (calling a software application) and *manual tasks* (signaling a user to fulfill a task outside the computer system) can be executed by a remote process management system if the required resources are (locally or remotely) available. However, rebinding resources requires a common understanding such as a shared organizational model or an ontology which is not part of BPMN. In consequence, the migration of BPMN processes relying on such (abstract) constructs requires an additional agreement on specific technologies to be used.

Furthermore, user interfaces for presenting process data and accepting the user's input are often process-specific and are deployed together with the process (e.g. an HTML form). The same is true for small code snippets which are carried along with the process. As such resources can be (physically) moved to the target system, migration of respective *user tasks* and *script tasks* is possible if the target system provides a compatible platform. However, the migration of arbitrary executable code (as with the *script task*) implies potential security risks and should thus be restricted to trustworthy partners only (cp. [14]).

Another restriction is required for messaging tasks: If *send task* and *receive task* belong to a pair of asynchronous communication (e.g. tasks 7 and 10 in Figure 2), such corresponding tasks must be executed on the same process engine in order to allow the answer message to be correctly delivered. This can be realized by applying a dynamic assignment strategy.

Finally, the specification of business rules is also outside the scope of BPMN. If business rule management systems are compatible and a *business rule task* is rebound to a local rule engine, it must be considered that the outcome may be different and thus may lead to a different control flow (cp. task 2 in Figure 2).

Table 1. Properties of throwing event types in BPMN 2.0

Type of event	Purpose of the event type	Latest point of subscription
top level start	instantiate process models	process deployment
event sub-process start	launch indepent sub-processes	start of surrounding (sub-)process
intermediate boundary	handle events in bounded flow objects	start of bounded flow object
intermediate catching	hold control flow until event occurs	control flow arrives at event

3.2 Gateways

A *gateway* is a control flow construct which determines forking and merging of alternative or parallel process paths depending on a set of (optional) conditions. With respect to the migration model, it is advantageous to consider a *gateway* to be a kind of atomic activity executed by the process engine itself. Now, the act of evaluating a condition can be assigned to a specific process engine (or a respective role) and thus it can be influenced which process engine should be responsible for its execution. Following this idea, a gateway is in the state *ready* if the process's control flow reaches the gateway, it is in the state *executing* while evaluating all of its conditions and is *executed* resp. *finished* if the outcome has been computed. Depending on the result, the states of the upcoming activities are set to *ready* or *skipped* (used by *dead path elimination*, cp. [14]). In consequence, migration is also possible during the execution of alternative or parallel paths.

3.3 Subprocesses

A *subprocess* represents a block activity containing an enclosed control flow which is executed in place of the subprocess. Therefore, it can be mapped to a complex activity in the migration model which enables migration at any time before, during or after its execution w.r.t. the state and migratability of enclosed elements. However, the description of the subprocess is required to be available at the target system. This is given if the calling process and the subprocess are specified within the same process description which is supported by both the graphical and the XML representation of BPMN (cp. task 3 in Figure 2).

3.4 Events

BPMN distinguishes events which are *thrown* by the process instance and events which are *caught* by the responsible process engine. Regarding migration, throwing an event is uncritical because it corresponds to a simple (atomic) activity which is performed by the process engine. In contrast, catching events involves the subscription for the respective events which are specified within the control flow, receiving the actual event instances, and starting the event handler (set of flow objects). In general, a (non-recurring) event can only be unsubscribed if the event has already been caught or the event-based control flow structure is finished. Considering the starting time and duration of a subscription for different event types (cp. Table 1), avoiding to split the procedure of *subscription*,

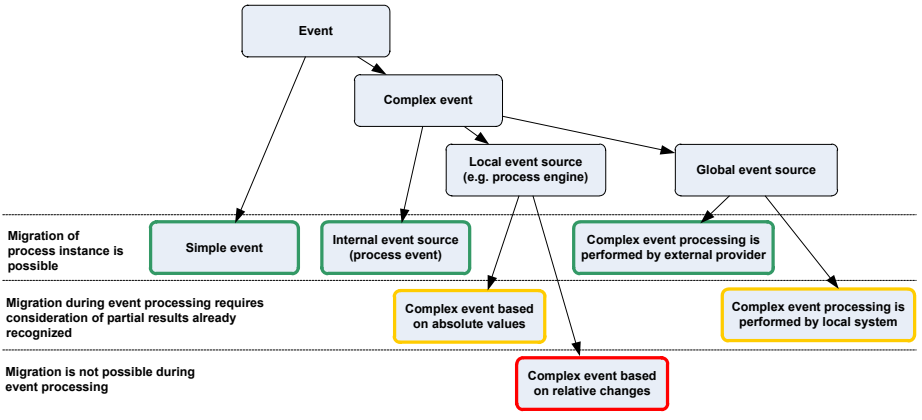


Fig. 3. Classification of events with respect to a process instance migration

receiving and *processing* of events would therefore prohibit migration for all event-based control flow structures. Thus, a more flexible solution is required.

Figure 3 shows an overview of respective migration possibilities resulting from general considerations on event processing. Accordingly, Table 2 evaluates the specific events defined in BPMN 2.0. Non-complex events (e.g. *error*, *compensation*, *message*) are non-critical, because such simple events are processed atomically and, in case of migration, they are handled either on the source or target engine. However, a rebinding of the event source may be required. Complex events which are composed of process internal events (e.g. *(parallel) multiple events* using error and message events) are also non-critical, since they can be regarded as structured activities (cp. subprocesses) and the occurrence of a sub-event can be tracked by means of the activity lifecycle state. If the process instance requires subscription of events with an *external source*, it is necessary to take a closer look at the type of event and its relationship to the local environment. Complex events processed by an external provider are recognized by the process engine as a simple event and therefore inherit the corresponding migratability. If complex event processing is performed by the local system, sub-events which are already detected on the current process engine would have to be transferred to the target engine. Other complex events using local event sources from the context of the process engine (e.g. an event concerning the temperature in the local environment) are feasible, if the complex event is based on absolute values and sub-events are transferred on migration. However, in case of relative changes, it is possible that events are triggered by migration. An example is a complex event which is thrown if the temperature rises up to 150% of its average value. If the ambient temperature of the source and target location of process execution essentially differ during migration, the event may be thrown although the temperature has not changed at both locations at all. Throwing the event would thus be semantically incorrect and has to be avoided.

Table 2. Evaluation of specific events in BPMN 2.0

Event name	Purpose	Compl.	Int.	Loc.	Rem.	Migration
Compensation	compensation of activities	–	✓	–	–	possible
Error	hierarchical processing of errors	–	✓	–	–	possible
Escalation	hierarchical processing of escalations	–	✓	–	–	possible
Terminate	terminates the current process instance	–	✓	–	–	possible
Link	connects control flow inside a process	–	✓	–	–	possible
Signal	signaling inside and across processes	–	✓	✓	–	possible[1]
Cancel	canceling transactions	–	✓	✓	–	possible[2]
Message	receiving a message	–	–	✓	–	possible[1,3]
Timer	wait for a specific time period/point	–	–	✓	–	possible[4]
Conditional	reference to arbitrary conditions	*	*	*	*	depends on source/complexity
Parallel/Multiple	requires all/one of multiple events	*	*	*	*	depends on used event types

[1] source rebinding possibly required
[2] see message event in case of external transactions
[3] the message event is locally triggered on receiving a (remote) message
[4] time periods can be replaced by an absolute point of time on activation

A reliable solution is to consider receiving events as (atomic resp. structured) activities which require the process engine to perform event processing. The activity is set to the *ready* state if the process's control flow reaches the event resp. the first activity of its bounding (resp. surrounding) scope. At this point of time, also the event source has to be subscribed at the latest. The event activity goes to the state *executing* at the time event processing is started, i.e. a (partial) event is received. Simple events have the character of an atomic activity and are *executed* and *finished* right after the event has been detected and the upcoming activity has been activated. Complex events consist of a rule for pattern recognition of multiple event parts along a time period and remain in the state *executing* until the event is completely detected and processing is ended. In that case, the event is set to *executed* resp. *finished* and upcoming activities are activated. In doing so, complex event processing cannot be interrupted by a migration of a process instance as this would lead to a loss of partial results of event detection and, possibly, incorrect results. In case of boundary events, there is also the option to set the event to the states *expired* resp. *terminated* if the execution of the scope is finished before the event has (completely) occurred. The process engine can now unsubscribe from the event source.

3.5 Pools and Swimlanes

BPMN 2.0 allows to express the collaboration of multiple resources and organizational units. The *swimlane* element represents the assignment of flow objects to roles or specific participants and thus represents a *resource level view* of a distributed business process. The *pool* element is a higher level construct to express which business unit or technical unit is responsible for the execution of a specific process partition and for calling the required resources. Pool elements can thus be mapped to the migration model as a pre-defined *assignment strategy* for migration targets based on design-time distribution requirements. In consequence, the distribution specified in the collaboration among pools (i.e. *message*

flows) can be used to represent the transfer of the process instance from one process engine to another. Process instance migration thus provides an interesting alternative interpretation of the distribution specified in BPMN's executable collaboration models. Basic strategies to also support a parallel execution of process partitions among different process engines can be found in [14].

4 Extension of the Activiti Process Engine

In order to estimate the efforts for extending an existing BPMN process engine with additional support for process instance migration, the open source Activiti process engine [2] was selected. In general, the procedure of process instance migration can be separated into four distinct phases:

1. Stop the process engine's execution of a process instance in a consistent state.
2. Extract and save runtime information, then delete the process instance or mark it as *transferring*.
3. Send process definition and runtime information to the target system.
4. Resume the transferred process instance on the target system.

Following this procedure, Activiti has been enhanced by an internal event concept in order to trigger a blocking event each time migration becomes possible, i.e. after an activity's execution has been finished. The new *migration handler* component handles such events by stopping the process execution if there is currently a demand for migration (i.e. based on user-defined migration strategies; phase 1). Otherwise, the execution is continued regularly. In the second phase, process instance data is retrieved from the process engine and is transformed according to the generic migration model. Activiti uses a relational database to store runtime information of processes in a proprietary schema, which is used by the migration handler in order to create a system-independent XML file containing the migration data. In phase three, this file and the (original) BPMN process model are transferred to another migration-aware (Activiti) BPMN engine by using a well-defined Web Service interface. After successful transmission, the target engine subscribes for local and remote events which are in the state *ready* and, subsequently, the source engine can unsubscribe for these events and delete the interrupted process instance. Finally, in the fourth phase, the transferred process instance is resumed. In Activiti, this is done by creating a new process instance and by applying the current state from the XML migration data.

The experiences with the prototype show, that, dependent on the process engine's architecture, deep intrusions in the engine's code may be necessary in order to gather the current process state and influence the process execution. However, the engineering effort for extending an existing BPMN process engine has to be performed only once in order to migrate arbitrary process instances automatically. Compared to a physical fragmentation which requires modification and deployment of each individual process model, this engineering effort is feasible – especially if many process instances have to be distributed individually.

5 Conclusion

Although BPMN 2.0 is developed as a process interchange format, the migration of running BPMN process instances holds several challenges. First, a meta model for process instance data is not part of BPMN. Second, the process description can be defined in a rather abstract way which requires rebinding of activities to new execution environments and resources. Based on an existing approach, this paper has thus proposed a mapping of BPMN 2.0 elements to a generic migration model and has discussed benefits and restrictions of the runtime migration of BPMN process instances. It shows that migration of BPMN process instances is – in general – possible. In order to ensure the original semantic of execution, migration has to be limited when tasks (including complex event processing) are currently executed. Due to its higher abstraction level, BPMN also allows for more flexibility in rebinding resources and events to the target system of migration. However, many relevant implementation details such as bindings to software applications, references to organizational models or the description and integration of user interfaces are not included within the BPMN standard and use additional (platform-dependent) instance data – which is still a challenge for ad-hoc process instance migration and has to be addressed in future work.

References

1. Atluri, V., et al.: A Decentralized Execution Model for Inter-organizational Workflows. Distrib. Parallel Databases 22(1), 55–83 (2007)
2. Baeyens, T., et al.: Activiti BPM Platform (2011), http://www.activiti.org/
3. Baresi, L., Maurino, A., Modafferi, S.: Workflow Partitioning in Mobile Information Systems. In: MOBIS 2004, pp. 93–106 (2004)
4. Bauer, T., Dadam, P.: Efficient Distributed Workflow Management Based on Variable Server Assignments. In: Wangler, B., Bergman, L.D. (eds.) CAiSE 2000. LNCS, vol. 1789, pp. 94–109. Springer, Heidelberg (2000)
5. Casati, F., Shan, M.C.: Dynamic and adaptive composition of e-services. Inf. Syst. 26(3), 143–163 (2001)
6. Cichocki, A., Rusinkiewicz, M.: Migrating Workflows. In: Advances in Workflow Management Systems and Interoperability, pp. 311–326. NATO (1997)
7. Fuggetta, A., Picco, G.P., Vigna, G.: Understanding Code Mobility. IEEE Transactions on Software Engineering 24(5), 342–361 (1998)
8. Montagut, F., Molva, R.: Enabling Pervasive Execution of Workflows. In: Collaborative Computing: Networking, Applications and Worksharing. IEEE (2005)
9. OMG: Business Process Model and Notation (BPMN), Version 2.0. Tech. rep., Object Management Group (OMG) (2011)
10. Schuler, C., Weber, R., Schuldt, H., Schek, H.J.: Scalable Peer-to-Peer Process Management - The OSIRIS Approach. In: ICWS, pp. 26–34 (2004)
11. Weske, M.: Business Process Management: Concepts, Languages, Architectures. Springer, Heidelberg (2007)
12. Wutke, D., Martin, D., Leymann, F.: A Method for Partitioning BPEL Processes for Decentralized Execution. In: ZEUS 2009, pp. 109–114. CEUR-WS.org (2009)
13. Zaplata, S., Kunze, C.P., Lamersdorf, W.: Context-based Cooperation in Mobile Business Environments. Bus. and Inf. Syst. Eng. (BISE) 2009(4) (October 2009)
14. Zaplata, S., et al.: Flexible Execution of Distributed Business Processes based on Process Instance Migration. Journal of System Integration (JSI) 1(3), 3–16 (2010)

Asynchronous Learning for Service Composition

Casandra Holotescu

Politehnica University of Timişoara
Dept. of Computer and Software Engineering
casandra@cs.upt.ro

Abstract. Correctness of system compositions is automatically ensured by using formal behavioural models of services. However, such models are not always provided. We present a model inference technique for black-box asynchronous services, that interleaves behavioural exploration and incremental model refinement. To save learning effort, only behaviour relevant to the desired system specification is explored. Compared to existing inference techniques that assume only controllable behaviour, our method addresses also uncontrollable events. Experimental results show that obtained models can be successfully used for a safe composition.

1 Introduction

The paradigm of service oriented applications relies on assembling third party web services together, so that the final system will satisfy some temporal property. This is done by building an adaptor/mediator – a service in-the-middle which properly coordinates the interactions between the other services towards the satisfaction of the desired composition goal. Several automatic, model-checking based techniques have been developed for mediator synthesis, such as [2,6,18]. All these techniques use specified formal models of the service behaviour. Unfortunately, providing formal specifications for third party services is still far from common industrial practice. One might often need to integrate a third party service with only partial, lightweight specifications. In this case, composition correctness can only be ensured by a preliminary model inference step.

For asynchronous services, which interact by sending/receiving messages, correct composition is also a control problem [21], where the controllable events are the receive events, while send events are uncontrollable. While a black-box synchronous, and thus, completely controllable service can have its model inferred by a variant of the Angluin algorithm [1], just as it is the case for black-box components in the works of Peled [12], Shahbaz [22], or Berg [3], things are different in the presence of uncontrollability. Angluin's algorithm assumes regular inference takes place as a series of membership and equivalence queries, answered to by an oracle [1], which for a black box component/service is the component/service itself. In the presence of uncontrollable events, however, such queries can no longer be generated as simple input sequences, or series of input sequences, since an uncontrollable transition can occur anytime.

G. Pallis et al. (Eds.): ICSOC 2011, LNCS 7221, pp. 76–86, 2012.

Our technique aims at composing systems that contain deterministic black-box services. To the best of our knowledge, it is the first model inference technique for asynchronous black-boxes, and, together with [22], one of the few that consider black-box model inference for compositional purposes.

Considering a desired temporal property, we delegate the task of coordinating the system to an intelligent adaptor, which monitors the services and controls their interactions by intercepting and forwarding their messages, thus actively exploring the behaviour relevant to the desired property.

For each black-box, we start with a most general model, which is incrementally refined each time it proves inconsistent with the run-time behaviour of the service, e.g. when a forwarded message is not accepted, etc. To save learning effort, only behaviour conforming to the system specification is explored. The final model obtained is a safe approximation of the black box behaviour, which can be used for adaptor synthesis. Building upon our previous work [16,17], this paper differs by an improved model refinement method, safety and termination proofs for presented algorithms, and an experimental evaluation of our technique.

2 Method

2.1 Assumptions

A desired system S is to be composed out of a set $\mathcal{W}$ of deterministic services, out of which n are black-boxes. Without loss of generality, we further on consider the case where all services in the system are incompletely specified, and $|\mathcal{W}| = n$.

Each black-box service WS_i is associated to a tentative model, which represents an approximation of its real behaviour. This model is described by a Büchi automaton, $U_i = \langle Q^i, q_0^i, Q_f^i, \Sigma^i, \delta^i \rangle$ where:

- Q^i is the state set and $q_0^i \in Q^i$ the initial state
- Σ^i is the event set of WS_i and $\Sigma = \bigcup \Sigma^i$ is the event set of the system
- $Q_f^i = Q^i$ is the set of accepting states
- $\delta^i : Q^i \times \Sigma^i \to \mathcal{P}(Q^i)$ is the transition function

An event σ in Σ is either a message send: $msg!$, or a receive: $msg?$. We denote by $\Sigma^i(q)$ the set of events σ for which $\delta^i(q, \sigma) \neq \emptyset$, with its two subsets $\Sigma_?^i(q)$ – the subset of controllable events, and $\Sigma_!^i(q)$ – the subset of uncontrollable events.

Considering a string of events $t_k = \sigma_0\sigma_1..\sigma_k$, and Σ_*^i the set of strings formed from events in Σ^i, then the extension δ_*^i to strings of events of the transition function δ^i is the string transition function $\delta_*^i : Q^i \times \Sigma_*^i \to Q^i$ such that $\delta_*^i(t_k) = \bigcup \delta^i(q, \sigma_k)$ for all $q \in \delta_*^i(t_{k-1})$, and $\delta_*^i(t_0) = \delta^i(q_0, \sigma_0)$.

If a trace $t \in \Sigma_*^i$ is observed at runtime, we denote this fact by $obs(t)$. If an event $\sigma \in \Sigma^i$ is observed from a state $q \in Q^i$, we denote by $obs'(\sigma, q)$.

We also assume for every service WS_i that the inner service scheduler is fair with respect to uncontrollable events. We assume a bounded fairness [8], considering a bound θ on the number of state visits for a single state q of the model as a fairness bound. This means that if a state q is reached for at least θ

times, which we denote by $visits(q)$, every uncontrollable event σ enabled in q is observed at least once: $\sigma \in \Sigma^i_!(q) \wedge visits(q) \geq \theta \longrightarrow obs'(\sigma, q)$.

Tentative model U_i approximates the real behaviour of WS_i, which we assume is precisely described by an unknown model R_i, also a Büchi automaton.

The desired system S must comply with a safety property Φ, described by a similar Büchi automaton. For S to comply with Φ, S must simulate Φ.

Let us also assume that $\forall i \leq n-1.ack!, rst? \in \Sigma^i$, where $ack!, rst?$ are special, auxiliary events. Event $ack!$ confirms a successful receive. Event $rst?$ forces the service to return to its initial state, from any state q. This assumption makes receive events externally observable and allows for reset. To ensure termination, we consider an upper bound m on the number of states of any service WS_i.

2.2 Behaviour Exploration

Let us denote by $U_\times$ the asynchronous product of the tentative models $U_\times = U_0 \times U_1 \times ... \times U_{n-1}$. As defined in [**?**], we consider a control point in $U_\times$ as a global state q_{cp} for which at least one of the outgoing controllable event sets $\Sigma^i_?(q_{cp})$ contains 2 or more receive events: $|\Sigma^i_?(q_{cp})| \geq 2$. This basically means that in state q_{cp} we can choose from at least two possible execution paths.

During the exploration process, the property automaton Φ is executed synchronously with the system model $U_\times$. If, from a state q, an uncontrollable event σ' occurs at runtime, and no transition triggered by σ' exists in Φ from the current state, then q is marked as a forbidden state. When the execution reaches a control point q_{cp}, a receive event $\sigma \in \Sigma^i_?(q_{cp})$ is enabled for execution if from the current state of Φ a transition on σ exists, and $\delta(q_{cp}, \sigma)$ does not contain forbidden states. If runtime observations report an inconsistency between U_i and the real behaviour of WS_i, U_i is refined. The execution is forced to end when:

- A forbidden state is reached.
- A cycle that respects the safety property is confirmed:
  - a cycle is found in the system model
  - the execution enters the cycle, and doesn't leave it for a number of λ transitions, during which the property automaton visits at least one accepting state – which ensures an infinite correct run.

The threshold $\lambda = m^n$ is the maximum state space size of the system. As stated in [12], for a black-box component with maximum m states, an accepted trace of the form uv^m, where u and v are traces in δ^i_*, and v^m represents the repetition of v for m times, is enough to prove the existence of a correct infinite run.

2.3 Trace Removal

The basic intention of the trace removal process is to refine the model by removing only the partial trace observed not to be accepted at runtime.

Let us consider a current execution trace $t_k = \sigma_0\sigma_1..\sigma_k$, observed at runtime: $obs(t_k)$. Let t_j, $j \leq k$, be a prefix of t_k such that a state q_j is reached after

performing the transitions triggered by t_j, and from the initial state q_0, q_j is uniquely reachable by t_j: $\{q_j\} = \delta^i_*(t_j)$, and, if q_j is part of a cycle in the model, t_j reaches q_j only once. If no such t_j exists, then q_j is the initial state.

Suppose now that after event σ_k, we try to enable controllable event σ_{k+1}, and fail. The trace $t_{k+1} = t_k.\sigma_{k+1}$ is thus missing from the behaviour of real service WS_i, therefore it has to be also removed from the behavioural model U_i, together with all traces prefixed by t_{k+1}, but without removing other possible strings of events. We denote the application of this removal process by $remove(t_{k+1})$.

Then, the refinement of U_i will have a number of $k - j + 1$ new states: $q'_j, q'_{j+1}, .., q'_{k+1}$ and the transition function δ^i is modified as following:

- $\delta'^i(q'_l, \sigma_{l+1}) = \{q'_{l+1}\}$, for $j \leq l < k$ (the new string of states corresponding to the unsatisfiable prefix).
  If $j = 0$, q'_0 is the new initial state of the model, otherwise $q'_j = q_j$.
- $\delta'^i(q'_k, \sigma_{k+1}) = \emptyset$ (infeasible transition)
- $\delta'^i(q'_l, \sigma) = \delta^i(q_l, \sigma)$, for $j \leq l \leq k$, $\sigma \neq \sigma_{l+1}$ (any transition not in prefix returns to initial model)
- $\delta'^i(q, \sigma) = \delta^i(q, \sigma)$, if $q \neq q'_j$ (all other model transitions are preserved)

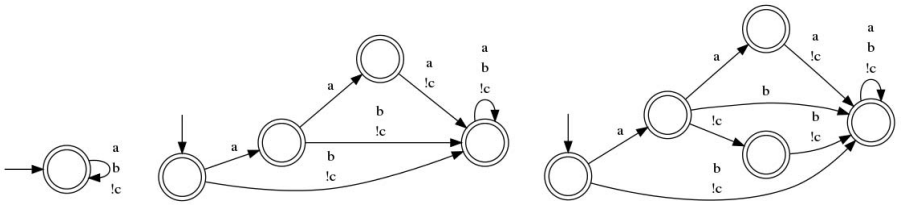


Fig. 1. Trace removal process: 1(a) initial tentative model, 1(b) refined model after removing trace $a?.a?.b?$, 1(c) model after second refinement: removing trace $a?.c!.a?$

After reaching the maximum state number m, the exploration and refinement process still continues until either the maximum number of executions γ, or the fixpoint condition has been reached. Refining the model, in this case, can no longer rely on the presented trace removal algorithm, since no new states can be created. So, in this case, the infeasible transitions on a controllable event σ are simply removed from their state of origin q, which we denote by $remove'(\sigma, q)$.

Expanding Forbidden Behaviour. When an erroneous trace $t_{bad} = \sigma_0\sigma_1..\sigma_k$ is observed, the model is refined and an unrolling of the error trace takes place, to isolate it from other traces with which might share a correct prefix. The purpose is to make it easier for the controller to disable the error trace on the final model.

The transformation is similar to the usual trace removal phase, with the difference that no transitions are removed from $\delta^i_*(t_{bad})$, but instead the new state $q'_p = \delta^i_*(t_{bad})$ is marked as forbidden: $bad(q'_p)$.

Final Pruning. When the exploration process ends, two final refinement actions are applied on a model U_i:

1. any controllable transitions yet unexplored are removed.
2. if a state was visited for a number of times that exceeds the fairness bound, the unobserved uncontrollable transitions are removed from that state

The reason for action 1 is that keeping infeasible controllable transitions in a model used to generate a controller for the system might lead to unsafe behaviour later on, in the composed system, since the adaptor/controller could try to enable an infeasible transition, fail, and thus reach an error state.

Action 2 is based on the fairness assumption, which we use to prune infeasible uncontrollable events, i.e. the uncontrollable events that didn't occurred within the fairness bound. If an uncontrollable event σ hasn't occurred from state q, after the service has been in state q for $nv \geq \theta$ times, then we conclude that σ never occurs from q. Else, all uncontrollable transitions from q rest unchanged.

Thus, each final model overapproximates the real uncontrollable behaviour and underapproximates the controllable one, which allows for a reliable adaptor synthesis an adaptor obtained for these models will also work on the real system.

2.4 Adaptor Synthesis

The adaptor for the system is computed from a controller enforcing the property Φ over the system plant. Consider a refined model of a service WS_i to be U'_i. Also, let $U'_\times$ be the asynchronous product of these refined models: $U'_\times = U'_0 \times U'_1 \times ... \times U'_{n-1}$. Thus, $U'_\times$ models the system plant. The computation of the controller, denoted by $Ctrl$, for the specification Φ, relies on the classical result of Ramadge and Wonham: $Ctrl = \mathbf{supcon}(U'_\times, \Phi)$, where **supcon**, described in [21], is a fixpoint procedure. All behaviours of the plant $U'_\times$ that do not violate Φ are allowed by $Ctrl$. Finally, adaptor A is obtained from controller $Ctrl$ by mirroring its event set.

2.5 Proofs

We prove below that the obtained adaptor is safe, and that the model learning algorithm eventually reaches its fixpoint termination condition.

Definition 1. *A controller $Ctrl$ that enforces a property Φ over a plant $\mathcal{P}$ is safe if all behaviour of controlled plant $Ctrl \| \mathcal{P}$ satisfies Φ: $\mathcal{L}(Ctrl \| \mathcal{P}) \subseteq \mathcal{L}(\Phi)$. An adaptor A is safe for a system $\mathcal{S}$ and a property Φ iff its corresponding controller $Ctrl$ is safe for the plant $\mathcal{S}$, and property Φ.*

Theorem 1. *Adaptor A, obtained for final models U'_i and property Φ, is safe.*

Proof. Suppose A is not safe $\longrightarrow$ controller $Ctrl$ over plant $U_\times$ is not safe. Thus: $\exists t \in \Sigma_*.\, t \in \mathcal{L}(Ctrl \| R_\times) \wedge t \notin \mathcal{L}(\Phi \| R_\times) \longrightarrow t \in \mathcal{L}(R_\times) \setminus \mathcal{L}(\Phi)$ Since $Ctrl = \mathbf{supcon}(U'_\times, \Phi) \longrightarrow \mathcal{L}(Ctrl || U'_\times) \subseteq \mathcal{L}(\Phi)$, we have $t \in \mathcal{L}(Ctrl \| (R_\times \setminus U'_\times))$.

Let $t = u.\sigma.v$, where $u \in \mathcal{L}(Ctrl||U'_\times)$ is a correct prefix, σ is the error-inducing event, and v represents the rest of the trace. Let $\sigma \in \Sigma^i$. Then: $u \in \mathcal{L}(Ctrl||U'_\times) \longrightarrow u.\sigma \notin \mathcal{L}(Ctrl||U'_\times)$. This implies $\sigma \in \Sigma^i_! \wedge P_i(u).\sigma \notin \mathcal{L}(U'_i)$.

The learning process has started with a most general model, so it means that σ was incorrectly removed from every set $\Sigma^i(q)$, where $q \in \delta^i_*(P_i(u))$.

Case 1. $P_i(u).\sigma$ removed by the trace removal procedure. If $|Q^i| < m$: $remove(t')$, $t' = t_k.\sigma_{k+1} \longrightarrow obs(t_k) \wedge \sigma_{k+1} \in \Sigma^i_?$. But $\sigma \in \Sigma^i_! \longrightarrow$ contradiction: $P_i(u).\sigma$ not removed. Else, if $|Q^i| = m$: $remove'(\sigma', q) \longrightarrow \sigma' \in \Sigma^i_?$. Also, $remove(P_i(u).\sigma) \longrightarrow P_i(u).\sigma = w'.\sigma'.w.\sigma \wedge \sigma' \in \Sigma^i_? \wedge remove'(\sigma', q)$ $\forall q \in \delta^i_*(w')$. But $remove(w'.\sigma') \longrightarrow remove(P_i(u))$. Since $obs(P_i(u))$, contradiction results.

Case 2. $P_i(u).\sigma$ removed in the transition pruning phase: $\sigma \in \Sigma^i_!(q) \wedge remove'(\sigma, q)$ implies $\neg obs(\sigma, q) \wedge visits(q) \geq \theta$. Since WS_i fair by bound θ: $\sigma \in \Sigma^i_!(q) \wedge visits(q) \geq \theta \longrightarrow obs(\sigma, q)$ Therefore, $\sigma \notin \Sigma^i_!(q)$, where $q \in \delta^i_*(P_i(u))$. But $obs(P_i(u).\sigma)$, thus $\sigma \in \Sigma^i_!(q)$, $\forall q \in \delta^i_*(P_i(u)) \longrightarrow$ contradiction.

From *Case* 1 and *Case* 2, we have that partial trace $P_i(u).\sigma \in \Sigma^i_*$ cannot be incorrectly removed from U'_i, therefore adaptor A is safe.

Theorem 2. *The fixpoint termination condition* $\forall i \leq n \wedge \forall q \in Q_i \wedge \forall \sigma \in \Sigma^i_?(q) \longrightarrow obs(\sigma, q) \vee bad(q')$, $\forall q' \in \delta^i(\sigma, q)$ *is eventually reached.*

Proof. Suppose the condition is never reached: $\exists i \leq n \wedge \exists q \in Q_i \wedge \exists \sigma \in \Sigma^i_?(q)$ *s.t.* $\neg obs(\sigma, q) \wedge \neg bad(q')$, $\forall q' \in \delta^i(\sigma, q)$. We know that $\sigma \in \Sigma^i_?(q) \longrightarrow \sigma = msg? \longrightarrow \exists j \leq n \wedge \exists \sigma' \in \Sigma^j_!$ *s.t.* $\sigma' = msg!$. Since all available messages are tried on, starting with those yet unconfirmed, then $\sigma \in \Sigma^i_?(q) \wedge \neg obs(\sigma, q) \longrightarrow \neg obs(\sigma', q)$. WS_j is fair under bound θ, therefore $\neg obs(\sigma', q') \wedge \sigma' \in \Sigma^j_!(q') \longrightarrow visits(q') \leq \theta$. Suppose $\{q'\} = \delta^j_*(w)$, where $w \in \Sigma^j_*$. Then, q' is reached for θ times in at most $\theta^{|w|}$ executions. Thus, it results that WS_j is not fair $\longrightarrow$ contradiction. Therefore, the termination condition is eventually reached.

3 Experimental Results and Discussion

We implemented our solution as the `BASYL` (Black-box `Asy`nchronous `L`earning) prototype. All experiments were run on an Acer Aspire 3820TG with a 2.26 GHz Intel Core i5 430M processor, 4 GB RAM memory and a 3 MB cache.

`BASYL` is written in Java and contains a monitoring component, that observes the events occurring at runtime and advances the service models accordingly, a decision logic component based on a bounded model-checker, that chooses the next controllable transition to be enabled, and a model refiner component, that modifies the approximate models whenever necessary. In order to link the high abstraction level at which `BASYL` works to a concrete technology, technology specific drivers are needed – for the case study presented below, we implemented such a driver for JMS.

To validate BASYL, we built an abstract version of one of the case studies presented in [24], using the Glassfish Server 3.1 implementation of Java Message Service for asynchronous message passing – it is worth noting here that using SOAP over JMS, although not a standard solution, is considered more reliable for building web services than SOAP over HTTP. We have chosen the case study of components ThinkTeam and CAD, but used only one CAD instance instead of two, and rewrote the coordination policy accordingly (see figure 2). For controller synthesis, the Supremica tool [19] was used. We assumed ThinkTeam as incompletely specified. Its alphabet contains 15 events, out of which 4 are uncontrollable, and the fairness bound θ was set to 15. We experimented with various maximum model sizes. The real model has 13 states and 22 transitions.

Because we safely approximate the real service behaviour, models that can lead to the synthesis of a safe adaptor can be obtained without reaching the refinement fixpoint. In the case studied, it took BASYL a number of 2 executions for $m = 5$, 90 for $m = 13$ and 290 for $m = 20$, to obtain models for which Supremica could synthesize system controllers (see Table 1). Since these models are not complete, but only safe approximations, the obtained controllers are more restrictive, and usually larger than the controller obtained for the real model.

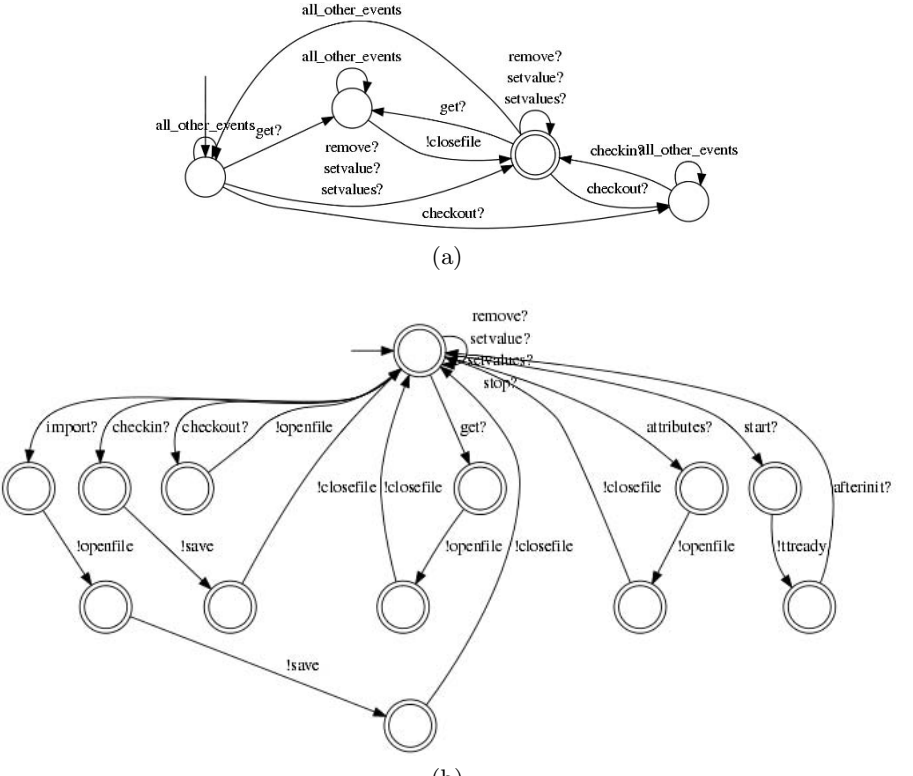


Fig. 2. Case study: 2(a) Property / Coordination policy, 2(b) Think Team real model

Table 1. Controller variation by max. model size and permissiveness

max. model size	min. ex. nr.	1st ctrl. size	1st ctrl tr.	γth ctrl size	γth ctrl tr.
5	2	7	58	28	204
13	90	56	380	84	577
20	290	126	841	133	913
real size = 13	**real model**	**56**	**247**	**56**	**247**

Due to the trace removal procedure, our technique does not lead to minimal models, as opposed to the Angluin algorithm. Therefore, we continue the exploration and refinement process even after the maximum state size is reached, until the fixpoint condition becomes true. However, the model is never guaranteed to be complete, i.e. equivalent to the real service model, due to the final pruning phase, while the Angluin algorithm obtains complete models within the state size limit. But what our algorithm does guarantee is obtaining safe approximate models, for which a generated controller is also safe, while the Angluin algorithm is not fit for learning systems with uncontrollable events, and would not produce safe approximations. Thus, our approach is specifically adapted to the reliable composition of asynchronous safety-critical systems. Also, depending on the system requirements, the system controller can be obtained in two ways:

- the first possible controller: after each model refinement, a controller synthesis phase takes place; if a controller can be obtained for the pruned model, the learning process stops. Thus, the learning process is shorter.
- the most permissive controller: since learned models become more precise with refinement, thus leading to increasingly more permissive controllers, the learning process only stops when the termination condition is reached.

Another limitation of our approach is its time complexity. As stated above, the cost of exploring behaviour by execution is greater than the cost of model checking, since paths are not explored simultaneously, but sequentially. Still, due to the assumption of fairness, each path can be eventually explored, if feasible.

If all states are control points, to obtain a trace of length λ a number of $O(|\Sigma|^\lambda)$ decisions are to be taken, which implies $O(|\Sigma|^\lambda)$ actual executions for an exhaustive, controllable exploration. This is, however, the large cost of black-box exploration. The black-box checking algorithm of Peled in [12] has a time complexity of $O(l^3|\Sigma|^l + l^3|\Sigma|^{m-1} + l^2pm)$ – where l is the size of an equivalent minimal automaton and p the size of the property automaton – to prove the correctness of the component, without addressing the controllability issue.

When exploring behaviour at runtime, we have to reset services each time we want to explore another path, and restart the run from the initial state. To reach again the original control point is difficult, due to the uncontrollable transitions in the system, which may steer the execution astray. To perform an efficient exploration, we need to reduce the number of necessary executions. This is why we explore only the paths that conform to the specification.

The number of runs needed to observe a trace depends on its controllability. If θ is the fairness bound, a trace t with p uncontrollable events needs up to θ^p runs to be observed. This is why we avoid classic querying: a query for membership of a trace t has a cost exponential in the number of its uncontrollable events.

4 Related Work

A classical reference in this area is the black-box checking technique of Peled et al. [12], that uses the Angluin algorithm [1] for model inference. An approximated model is proposed, verified, and compared to the real behaviour. Found differences are used to generate a new model, while counterexamples are validated by testing. However, their model only has input events, which highly simplifies the learning process, and their model underapproximates real behaviour. Also, in our case, verification, testing and model refinement are strongly interleaved, which allows us to obtain an useful model early, while their work has distinct such phases, aiming for an early confirmed counterexample.

Recent advances on dynamic model mining underline the critical significance of this domain. The works of Berg et al. [3], or the GK-tail algorithm developed by Lorenzoli et al. [14] focus on extracting extended finite state machines from execution traces. GK-tail uses inferred invariants and positive execution samples to extract EFSMs, while the method of Berg et al. uses an adaptation of the Angluin algorithm [1], thus querying for trace membership and model equivalence. In contrast, our approach learns the model on-the-fly, using both positive and negative samples, while always maintaining a safe approximation of the real service behaviour, instead of querying for equivalence.

Derived from the Angluin algorithm is also RALT, the work of Shahbaz [22], which can infer parametrized Mealy machines by adding invariant learning to regular inference, and studies a component behaviour first in isolation and then while in interaction with the rest of the system. Other automata learning tools are LearnLib [20], which also learns Mealy machines, and Libalf [5], which learns deterministic and even non-deterministic automata – both are algorithm libraries, containing various optimizations and extensions of the classic Angluin algorithm. However, none of these approaches deals with the issue of uncontrollable events, which would make membership and equivalence queries difficult.

In [23], Suman et al. describe a method to extract state models for black-box components under the form of finite state machines with guard conditions. It considers that a state is defined by the method invocations it accepts, and it discovers potential new states by invoking all active methods from the current state. The work of Dallmeier et al. in [7] relies on test case generation to systematically extend the execution space and thus obtain a better behavioural model by dynamic specification mining. Also, in [9], Ghezzi et al. infer behavioural models of black-box components that embody data abstractions. These techniques work with synchronous method calls, while our approach addresses asynchronous message exchange, which involves the issue of uncontrollable events. Also, our technique learns the model not in isolation, but with respect to both the desired property and feasible interactions with the other services in the system.

In the area of web services, the work of Bertolino et al. [4] and Cavallaro et al. [13] is related to ours. While [4] uses the WSDL interface to extract possible dependencies between the I/O data, and then validates them through testing, thus obtaining the behavioural protocol, the work in [13] takes this technique further, and develops a method for synthesizing web service adaptors that enable a correct

interoperation when an old service is replaced by a new one. However, the web services in [4,13] are stateless, while our approach addresses stateful black-box services. While their work starts from a data-dependency perspective, ours explores the language of the service and its controllability, regarding data at a higher level of abstraction, as passed messages. Therefore, we consider the two approaches complementary.

Grabe et al. [10] develop a technique for testing asynchronous black-box components, by using an executable interface specification. Their method addresses the difference between interacting under component control or under environment control, by describing the component behaviour in an assumption-commitment style. This work is closely related to ours, as we also test asynchronous black-boxes, and we also use a variant of an executable specification when we synchronize the property automaton with monitored services. However, their method doesn't infer a model for the tested component.

Another related method is [15] by Păsăreanu et al., which relies on environmental assumption generation when verifying a software component against a property. Their approach is based on the work of de Alfaro and Henzinger [11] stating that two components are compatible if there exists an environment that enables them to correctly work together. This divide-and-conquer technique analyzes components separately to obtain for each the weakest environment needed for the property to hold. By building a system controller, our approach also creates such an environment. However, while in [15] the analyzed component is well specified, our approach addresses systems that contain black-box services, whose behaviour and controllability must be understood before building an adaptor.

5 Conclusion

We presented a method to automatically compose a system with black-box asynchronous services. `BASYL` infers safe behavioural models for the black-box services by incrementally refining a tentative model. To learn only behaviour useful to the composition goal, the system is controlled by an intelligent adaptor, which guides the run towards the satisfaction of the system specification. The main contribution of our method is enabling a safe composition for systems with asynchronous black box services, by inferring safe approximations of the real service behaviour. Experiments performed show that `BASYL` can obtain models precise enough for controller synthesis, and can obtain them early in the exploration process.

Acknowledgments. We are grateful to Marius Minea and Ioana Bercea for consistent and useful feedback. This work was partially supported by the European FP7-ICT-2009-5 project no. 257876 SPaCIoS ("Secure Provision and Consumption in the Internet of Services"), by the Politehnica University of Timişoara and by the strategic grant POSDRU 6/1.5/S/13 (2008) of the Ministry of Labour, Family and Social Protection, Romania, co-financed by the European Social Fund: Investing in People.

References

1. Angluin, D.: Learning regular sets from queries and counterexamples. Inform. and Computation (1987)
2. Berardi, D., et al.: Automatic service composition via simulation. Int. J. of Foundations of Computer Science (2009)
3. Berg, T., Jonsson, B., Raffelt, H.: Regular Inference for State Machines with Parameters. In: Baresi, L., Heckel, R. (eds.) FASE 2006. LNCS, vol. 3922, pp. 107–121. Springer, Heidelberg (2006)
4. Bertolino, A., Inverardi, P., Pelliccione, P., Tivoli, M.: Automatic Synthesis of Behaviour Protocols for Composable Web-Services. In: ESEC/FSE 2009 (2009)
5. Bollig, B., Katoen, J.-P., Kern, C., Leucker, M., Neider, D., Piegdon, D.R.: `libalf`: The Automata Learning Framework. In: Touili, T., Cook, B., Jackson, P. (eds.) CAV 2010. LNCS, vol. 6174, pp. 360–364. Springer, Heidelberg (2010)
6. Calvanese, D., et al.: Automatic Service Composition and Synthesis: the Roman Model. IEEE Data Eng. Bull. (2008)
7. Dallmeier, V., et al.: Generating Test Cases for Specification Mining. In: ISSTA 2010 (2010)
8. Dershowitz, N., Jayasimha, D.N., Park, S.: Bounded Fairness. In: Dershowitz, N. (ed.) Verification: Theory and Practice. LNCS, vol. 2772, pp. 304–317. Springer, Heidelberg (2004)
9. Ghezzi, C., Mocci, A., Monga, M.: Synthesizing Intentional Behavior Models by Graph Transformation. In: ICSE 2009 (2009)
10. Grabe, I., Kyas, M., Steffen, M., Torjusen, A.B.: Executable Interface Specifications for Testing Asynchronous Creol Components. In: Arbab, F., Sirjani, M. (eds.) FSEN 2009. LNCS, vol. 5961, pp. 324–339. Springer, Heidelberg (2010)
11. de Alfaro, L., Henzinger, T.A.: Interface automata. In: ESEC/FSE-9 2001 (2001)
12. Peled, D., Vardi, M.Y.: Black box checking. In: FORTE/PSTV 1999 (1999)
13. Cavallaro, L., et al.: Synthesizing adapters for conversational web-services from their WSDL interface. In: SEAMS 2010 (2010)
14. Lorenzoli, D., Mariani, L., Pezzè, M.: Automatic Generation of Software Behavioral Models. In: ICSE 2008 (2008)
15. Păsăreanu, C., Giannakopoulou, D., et al.: Learning to divide and conquer: applying the L* algorithm to automate assume-guarantee reasoning. In: FMSD 2008 (2008)
16. Holotescu, C.: Controlling the Unknown. In: FoVeOOS 2010 (2010)
17. Holotescu, C.: Black-Box Composition: a Dynamic Approach. In: SAVCBS 2010 (2010)
18. Marconi, A., et al.: Automated Composition of Web Services: the ASTRO Approach. IEEE Data Eng. Bull. (2008)
19. Åkesson, K., Fabian, M., et al.: Supremica: an integrated environment for verification, synthesis and simulation of discrete event systems. In: WODES 2006 (2006)
20. Raffelt, H., Steffen, B., Margaria, T.: Dynamic Testing Via Automata Learning. In: Yorav, K. (ed.) HVC 2007. LNCS, vol. 4899, pp. 136–152. Springer, Heidelberg (2008)
21. Ramadge, P., Wonham, W.: The control of discrete event systems. Proc. of the IEEE 77(1) (1989)
22. Shahbaz, M.: Reverse Engineering Enhanced State Models of Black Box Software Components to support Integration Testing. Ph.D Thesis (2008)
23. Suman, R.R., et al.: Extracting State Models for Black-Box Software Components. J. Obj. Tech. (2010)
24. Tivoli, M.: An architectural approach to the automatic composition and adaptation of software components. Ph.D Thesis (2005)

Engineering Energy-Aware Web Services toward Dynamically-Green Computing

Peter Bartalos and M. Brian Blake

Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, Indiana, USA
{peter.bartalos.1,m.brian.blake}@nd.edu

Abstract. With the emergence of commodity computing environments (i.e. clouds), information technology (IT) infrastructure providers are creating data centers in distributed geographical regions. Since geographic regions have different costs and demands on their local power grids, cloud computing infrastructures will require innovative management procedures to ensure energy-efficiency that spans multiple regions. Macro-level measurement of energy consumption that focuses on the individual servers does not have the dynamism to respond to situations where domain-specific software services are migrated to different data centers in varying regions. Next-generation models will have to understand the impact on power consumption for a particular software application or software service, at a micro-level. A challenge to this approach is to develop a prediction of energy conservation a priori. In this work, we discuss the challenges for measuring the power consumption of an individual web service. We discuss the challenges of determining the power consumption profile of a web service each time it is migrated to a new server and the training procedure of the power model. This potentially promotes creating a dynamically-green cloud infrastructure.

Keywords: Energy-awareness, web service, service-oriented software engineering, green web service.

1 Introduction

By applying clean software interfaces to human-based capabilities or legacy software systems [11], web services are modular, network accessible software applications that promote the open sharing of domain-specific capabilities across organizations. With the current priority on sustainability, attention must be placed on the energy-efficiency of IT assets in all organizations. While the large majority of work in energy-awareness concentrates on measuring and repurposing hardware resources, the aim of this paper is to understand how to allocate the specific software service that is most energy-efficient. This paper provides insight into the problem of determining the power consumed when processing a particular web service operation. This work leverages the fact that one web service can provide multiple operations while also potentially being replicated on multiple

G. Pallis et al. (Eds.): ICSOC 2011, LNCS 7221, pp. 87–96, 2012.

servers. More specifically with respect to energy usage, the *same* operation is accessible at multiple places; however, invoking the *same* operation on *different* servers might result in varying degrees of power consumption. It is also important to realize, that the *same* operation invoked on the *same* server, but at a *different* time, also may result in a *different* power consumption. In other words, the power consumption changes over time and across regions. This is mainly caused by the fact that invoking the same operation during distinct states of the server, mainly characterized by the utilization of its hardware resources, does not result in performing exactly the same computation. This means that the power consumed when processing a particular web service operation request is not invariant according to time and requires *dynamic* determination.

Several factors must be considered when determining the power consumed by web services. The power estimation model must provide power estimations relative to distinct possible states of the server. Thus, if the state of the server at the time the request is to be processed is known, we can provide accurate consumption estimation for a particular web service on that server. Since the future state of the server is unpredictable, this approach assumes that the server state changes negligibly in the short period of time that it takes to execute a typical web service. We anticipate that our approach will be effective for providing *dynamic* power estimation information to *environmentally-aware* web service management systems that dynamically discover and compose web services across clouds [1,2,12]. This approach will promote effect decision-making when selecting the particular instance of the service (service running on a concrete server) that consumes the *least energy* of multiple redundant options. This approach that we call *green web services* is an inevitable step towards the realization of sustainable cloud environments.

Our work addresses the following research questions:

- *What are the challenges of evaluating the power consumption of web services?*
- *Which hardware conditions, as measure by system performance monitors or counters, should be used and how should the data be monitored and collected?*
- *How is a model for a specific web service achieved? How can a model be devised through a training process?*

2 Power Modeling Background and Related Work

State-of-the-art studies have demonstrated that relatively precise computer power consumption estimations can be derived from the readily-available hardware performance system measurement tools. In this context, the challenge is to: 1) choose the appropriate counters that most effectively capture the state of the computer, 2) define the mathematical model calculating the estimated power from the computer state, and 3) define the power model training process.

There are currently several power estimation models in related literature [3,5,6,10,9,4,8]. Through these models, an estimation of the power consumption

of the server, under various workloads, based on readily available system measurement tools can be created. Consistently across the related work, the power estimation models leverage system measurement tools or counters that monitor the hardware conditions on the machines for which they reside. These counters measure conditions such as the *usage of different CPU instructions*, *various memory access operations*, and *distinct hard drive access*. The power estimation model is represented by a function of these attributes. This function is usually a linear combination of the attributes, $X_1, \ldots, X_n$, with an additional constant representing the idle power consumption: $P = P_{idle} + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n$, i.e. the model is represented by the coefficients, β, of the attributes with additional constant for the idle power. The values of these coefficients are determined during a training process by reggression.

Related Work. The authors of [9] present an approach for run-time modeling and estimation of operating system power consumption. They focus on instantaneous estimation of the power consumed when executing OS-intensive workloads (file management, compilation, database operations). One of the main results of the study is the observation of a high correlation between the IPC and power. In [4], the authors present a power estimation model for the Intel XScale®PXA255 processors. The approach exploits the insight into the internal architecture and operation of the processor. It achieves accuracy within an average of 4% error using five processor performance counters. The authors of [8] focus on power consumption estimation of virtual machines running on the same physical machine. Their solution is implemented by their customized tool, *Joulemeter*. It uses a similar power model as the other mentioned approaches. However, their motivation to use performance counters to predict power consumption is different. Direct measurement of the power is possible only for a physical device. If multiple virtual machines are running on one physical machine, it is not possible to measure the power consumption of the virtual machines separately. Since the performance counters can be monitored separately for each virtual machine, they attempt to segregate the power consumption. The power estimation of Joulemeter achieves accuracy, with errors within 0.4W - 2.4W.

Our Approach. Our experiments were performed using a model represented as a linear combination of the following attributes: number of received/sent packets, instructions executed, CPU cycles (the counter increments when the core clock signal is not halted, i.e. it is varying according to the changing load), *IPC* (i.e. instruction divided by cycles), percentage of *non-idle CPU* time, and *last level cache misses*. The values of the weights (i.e. beta vector) are acquired by performing a *training stage*, during which, samples of the measured attribute values and the resulting power are collected. The power value is retrieved using a physical power meter that is connected to the server. After the samples are collected, a regression method based on least squares is used to evaluate the power weights. Note that the power meter is only required during the training stage. This paper presents an insight to the specific problems related to the power estimation of web services. We evaluate different training processes through experiments

and show which approach results in a model estimating the instantaneous power (measured in Watts) during web service workloads the best. Note that in the case of web service workloads, the power model can only be used to estimate the total power. To estimate the power consumption (measured in Watt hours) of one particular request contributing to the workload, the total power must be split and the time dimension must be considered. Our preliminary approach performs this according to the number of actually processed requests and the execution time.

3 Web Service Power Estimation Challenges

Isolating the Specific Web Service. To determine the power consumption of a specific web service, the process should clearly segregate the impact of a particular request. In general operations, web servers continuously process multiple parallel web service requests. Each particular request might correspond to the execution of distinct web service operation or operations (when considering a composition service). Web services exploit different hardware resources, such as CPU, memory, hard drives. These resources all represent some part of the total power consumed as a result of the web service processing. Since the resources are *shared* and exploited in *parallel* by multiple requests, it is not feasible to determine the portion of the consumed power related to each of the requests. As such, a physical power meter cannot be used to measure the power consumed by one particular request. The power meter, generally, can only measure the overall power consumption of the server.

Web Service Executions Are Inherently Short in Time Duration. Since several web services execute in relatively short time, e.g. in milliseconds, the general power meters do not effectively capture accurate measurements at this scale. Furthermore, synchronizing the measurement with the processing of the request by the server is a challenge. Thus, isolating the power consumption of web service requests requires a custom model informed by the SOA paradigm. This model must estimate the power consumed when processing a particular web service operation request on a concrete server at a given moment.

Power Consumption Depends on the Actual Server State. Our preliminary experiments showed strong dependence between the actual server state and the power consumed to process a web service request. Significant differences are observed even when the same web service operation is executed with the same inputs. The most influencing factor, describing the server state, is the CPU load. Our results show that the nominal power consumption of one request is much higher when the server is under-utilized. As the utilization rises, the differences are lower. In our experiments, the nominal power consumption, while the server was under-utilized, was in average 3 times higher than the consumption when the utilization was high. However, at very high utilization rates, the consumption rised again. Thus, the same results when measuring the power consumption are only anticipated if the same circumstances are guaranteed.

Relationship between Input Data and Service Computation Is Unpredictable. The knowledge of the inputs in advance is limited and in most of the practical scenarios this information is not available. In general we can expect a relation between the size, and the structure of the inputs and the complexity of the computation. Considering the (de-)serialization of the I/O into/from messages, large messages and more complex structures are more computationally demanding. The dependence at the execution phase does not necessarily have to hold. However, for many types of web services there is an obvious dependence. For example, in the case of a web service sorting numbers, there is a defined relation between the size of the input sequence and the amount of required computation affecting the power consumption. As a result, for some web services, an effective approach is to determine the power estimation by referencing a value calculated when the service was invoked using a representative set of inputs. This value might be defined as the average, or maximum - similar to the response time determined when evaluating the QoS characteristics of the service.

4 Practical Problems When Building Power Estimation Models

To build the power model, data must be collected using distinct software and/or hardware components. To train the model correctly, the data collection must be synchronized. Misalignment of data samples introduces errors in the power estimation, so a precise sampling with respect to their alignment is a challenge. The following types of data must be collected: 1) hardware performance counters, 2) web service execution statistics, and 3) power measurements. There are several software tools and devices that provide access and facilitate the measurement of the data required to build the power model for the server.

Monitoring System/Hardware Conditions. Measuring system performance is possible due to built-in registries holding data related to specific events related to the hardware, e.g. number of instructions, cycles, and last level cache misses in CPU. The number of these registries on a CPU, i.e. counters incrementing when the event occurs, is limited. Each particular processor model provides a set of events, which might be measured by the programmable counter. Using these, it is possible to develop various on-demand monitors. On some processors, the length of the registries storing performance data is too low. This causes frequent overflows which then must be handled at higher application level. Some performance statistics, such as network and hard drive traffic, are monitored at the operating system (OS) level. Thus, OS specific tools must be used to access the statistics. In our experiments, we used the *Dell PowerEdge SC 1430* server, with two *Intel Xeon 5130* dual core processors, running *Ubuntu 10.04*. The network traffic and the percentage of the non-idle CPU time were collected using a *Libstatgrab library*, written in C. The CPU instructions, cycles, and last level cache misses were measured by *Intel Performance Monitor Counter*, written in

C++. These applications showed high reliability during the experiments. Other useful tools include:

- *Libstatgrab library* https://launchpad.net/ubuntu/+source/libstatgrab/0.17-0ubuntu1
- *Microsoft Windows Performance monitor* http://technet.microsoft.com/en-us/library/cc749249.aspx
- *Windows Management Instrumentation* http://msdn.microsoft.com/en-us/library/windows/desktop/aa394582%28v=vs.85%29.aspx
- *V-Tune Amplifier XE* http://software.intel.com/en-us/articles/intel-vtune-amplifier-xe/
- *Microsoft Visual studio: Premium and Ultimate version.* http://www.microsoft.com/visualstudio/en-us

Web Service Execution Statistics. Web service related statistics, such as the number of requests and the corresponding response time, are natively supported by different web servers. This also means that the access to these statistics is platform dependent. In Java environment the *servlet filters* facilitate the collection of data. A filter can be implemented such that it triggers custom code each time the server receives a request and when it sends back the response to the client. On ASP.NET platform, *intercepting filters* are equivalent. In our work, the Java servlet filters were used.

Power Measurements. Measuring the power consumed by a computer usually requires an external physical power meter. Although, hardware-level measurement of the computer power consumption is available on some platforms, e.g. [7], these measurements generally do not occur a priori. There are several attainable power meters that provide an interface to communicate with computers. The communication is usually based on USB or Internet but is also accompanied by a noticeable delay. Accessing the data from a device requires custom applications. The device itself and the related applications generally do not provide the data at defined moments. Thus, synchronization with other measurements performed on a computer is limited. In our work, we use the *Watts UP .net* power meter connected via USB. To access its data, we enhance the linux application available at (`https://www.wattsupmeters.com`), written in C.

Synchronizing All Measurements. It is clear from the previous discussion that collecting all the necessary data requires multiple applications running concurrently. The synchronization of these diverse applications, running in parallel, presents a big technical challenge. In our work, a central Java application manages the overall process by invoking the underlying applications in parallel. The inter-application control is performed using *Java Native Interface* in the case of PCM, and using the `Runtime.exec()` method for the rest of the application. Since communication with the Watts UP device is accompanied by a non-negligible delay, the central application synchronizes all the other applications according to a signal received from the application handling the Watts UP device. The central application collects a defined number of samples, each of them

taken during a specified time period. Ultimately, the applications are synchronized only at the beginning of the data collection to limit the inter-application communication. Our experiments show, that the applications remain sufficiently synchronized even if the specified time period is 100 seconds. The misalignment at the end of the collection is few milliseconds.

5 Training the Power Model

Our approach to generate a power model, while addressing the specific requirements of web service workloads, is based on a controlled simulation of web service requests. The process recreates a variety of circumstances under which the server is monitored and performance measurements are captured. We developed a set of 6 different synthetic web services, each having 3 - 4 operations (20 in total), which are invoked during the simulation. Each web service tends to utilize distinct hardware resources, i.e. CPU, memory, hard drive, or a varied combination.

The services used to test the model were created by wrapping operations available in a mathematical library *Apache commons library* `http://commons.apache.org/`. We implemented 11 distinct web service operations which varied in the nature of their computation. Their execution time varies from a few milliseconds up to 10s of milliseconds. We created two testsets using these web services. In the first case, the service workload was generated using the same operation over a defined period of time. Subsequently, another operation was executed in the same fashion. This environment emulates a server that executes a smaller, limited set of web services at a time. The second testset was created by randomly selecting the operation before every invocation. In both cases, we simulated multiple clients independently invoking the web services concurrently. While performing the workloads, we collected performance counter data to be used as inputs for the model generation.

By running different configurations of our test environment and varying the number of clients (running on the same network), we emulated web service workloads. In determining how to invoke the synthetic web services for training the power model, we tested two simulation strategies. In our first simulation strategy, the client randomly selects the requested operation before every invocation, i.e. the requested operation changes over time. In the case of multiple clients, this may result in parallel execution of different operations. In the second strategy, the client continuously invokes the same web service operation, specified by the training manager. The training manager starts the clients, collects the required number of samples, and terminates the clients. In the case of the second strategy, the training manager also specifies the specific web service operation thus cycling through all the operations.

In our experiments, we tested the power model using three public benchmarks: *CPUburn* utilizing the CPU - the *burnBX* command was used `http://linux.softpedia.com/get/System/Diagnostics/cpuburn-1407.shtml`, *Tiobench* for hard drive IO operations `http://linux.die.net/man/1/tiobench`, *MBW* testing the memory `http://manpages.ubuntu.com/manpages/lucid/man1/`

mbw.1.html . The power model (built using the benchmark workloads) achieved a precision within a 4.84% error. It was less accurate than the models built using the synthetic services-based workloads. We anticipate, that this difference in error was a result of the benchmark workloads not incorporating web service-specific computations, e.g. serialization of the I/O and the overall management of the request processing.

An important issue when training the power model is determining the number of samples to collect. In general, more samples result in a more accurate model. We collected 3200 samples, sampled every second, for both strategies. Tab.1 presents the mean relative error of the power model according to the number of samples. The table illustrates the error separately for all of the test web service operations, the average through all operations, and the error when the operations were selected randomly. The different columns represent what portion of the 3200 collected samples were used to train the model (i.e. 10%, or 100%), for the two strategies S_1 and S_2. The mean error of the model built from 320 samples while performing strategy S_1 is 3.86%. S_2 achieves better results. The error in this case is 2.49%. When using all the 3200 samples to train the model, S_1 achieves mean error 1.67%. S_2 results almost the same error, 1.66%. Considering our server, this error corresponds approximately to 3.3 Watts. The bottom line of Tab.1 shows that estimating the power while the service workload is made by random operations is more accurate. Considering the number of required samples, we can conclude that S_1 requires more samples to achieve favorable accuracy. When the number of samples is sufficiently high, it achieves similar results as S_2. In this case, S_1 showed to be more accurate for the randomly selected service workload. In the case that fewer samples are available for training, S_2 achieves better results. Fig.1 presents the error of the power model built using strategy S_2 with 3200 samples. Each point of the graph represents one pair of the measured values and its corresponding estimation retrieved from the model. Ideally, the points should lie on a line y = x. The points above the line are overestimations of the power and the points below the line represent underestimation.

Note that fewer samples do not necessarily cause higher error. Consider for example results for the *TestAddhoc* web service operations and strategy S_2. The error is lower in the case when fewer samples were used. This means that the model built using the selected 10% of the samples better characterizes the power consumption while executing *TestAddhoc*. However, for several other operations, this leads to lower accuracy, i.e. the model is less universal.

We also experimented with the training process executing locally on the server. The experiments demonstrated that the models built while the service workload was made by remote clients are more stable. It achieves similar results independently on the workload. The model, using our first strategy, built while executing the clients locally achieved low accuracy within 3.0% error. In the case of randomly selected services, the accuracy within 1.0% error, was better than in the case of remote clients. We can conclude that the models built with remote clients are more universal.

Table 1. Power estimation model error

	S_1 **10%**	S_2 **10%**	S_1 **100%**	S_2 **100%**
TestSort	5.28	0.49	1.65	0.57
TestAddhoc	6.27	0.53	1.69	0.64
TestAxisAngle	6.21	0.47	1.59	0.56
TestCircleFit	1.95	5.67	3.48	4.09
TestExactIntegration	4.24	0.59	1.54	0.60
TestLongly	0.82	4.16	1.67	2.56
TestLonglySpearman	2.28	4.10	0.85	2.20
TestError	2.39	1.38	1.92	1.07
TestQRColumnPermut	5.18	1.58	2.87	2.53
Testwave	4.65	3.08	0.85	1.00
Testplane	3.99	3.84	0.53	1.77
Average	3.86	2.49	1.67	1.66
Random selection	2.95	1.5	1.25	1.3

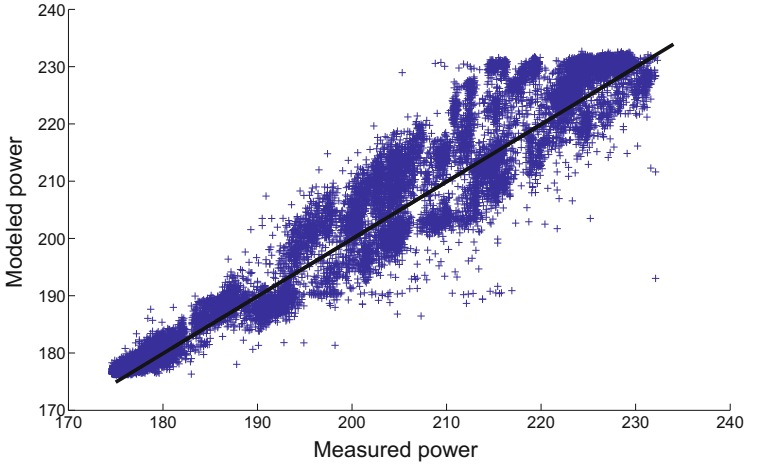


Fig. 1. Power estimation model error

6 Conclusions

This paper explores server-side power consumption of web services. We discuss the conceptual and practical problems when estimating the power required during web service workloads. The approach is based on a model that derives the power estimation from a variety of computer hardware conditions. We present an evaluative body of knowledge of the training process, which is a critical part of building the power model. We performed several experiments to investigate what aspects affect the ability of the model to estimate the power consumed by a particular web service. Our experiments demonstrated that the most accurate universal model is built when the workload during the training is made by remote clients. We considered two strategies to create service workloads. In first

strategy, one operation is selected to be executed for a defined period of time. In the second strategy, the operation is randomly selected before every invocation. Assuming that a sufficient number of samples are recorded during the training, there is no significant difference between the two strategies. The strategy executing one web service at a time showed more accurate results even when the model is built from fewer samples.

Acknowledgments. The authors would like to thank Mr. Chris Ketant of Rochester Institute of Technology for insights and his development of several of the synthetic web services. The authors would like recognize the interaction with Dr. Sekou Remy in comparing our regression approaches to relevant neural network approaches for training the greenness web services. This work benefited greatly from discussions with Tanya Salyers, Department of Mathematics, University of Notre Dame and Dr. Roman Dementiev from Intel. This project was partially supported by NSF Award Number 0512610.

References

1. Bartalos, P., Bielikova, M.: Qos aware semantic web service composition approach considering pre/postconditions. In: IEEE Int. Conf. on Web Services, pp. 345–352 (2010)
2. Bartalos, P., Bielikova, M.: Automatic dynamic web service composition: A survey and problem formalization. Computing and Informatics 30(4), 793–827 (2011)
3. Bircher, W., John, L.: Complete system power estimation: A trickle-down approach based on performance events. In: IEEE International Symposium on Performance Analysis of Systems Software, pp. 158–168 (April 2007)
4. Contreras, G., Martonosi, M.: Power prediction for intel XScaleR processors using performance monitoring unit events. In: Int. Symposium on Low Power Electronics and Design 2005, pp. 221–226. ACM (2005)
5. Economou, D., Rivoire, S., Kozyrakis, C.: Full-system power analysis and mod eling for server environments. In: Workshop on Modeling Benchmarking and Simulation (2006)
6. Fan, X., Dietrich Weber, W., Barroso, L.A.: Power provisioning for a warehouse-sized computer. In: International Symposium on Computer Architecture (2007)
7. Jenne, J., Nijhawan, V., Hormuth, R.: Dell energy smart architecture (desa) for 11g rack and tower servers (2009), http://www.dell.com
8. Kansal, A., Zhao, F., Liu, J., Kothari, N., Bhattacharya, A.A.: Virtual machine power metering and provisioning. In: 1st ACM Symposium on Cloud Computing, SoCC 2010, pp. 39–50. ACM, New York (2010)
9. Li, T., John, L.K.: Run-time modeling and estimation of operating system power consumption. SIGMETRICS Perform. Eval. Rev. 31, 160–171 (2003)
10. Rivoire, S., Ranganathan, P., Kozyrakis, C.: A comparison of high-level full-system power models. In: Conference on Power Aware Computing and Systems, HotPower 2008, p. 3. USENIX Association, Berkeley (2008)
11. Schall, D., Dustdar, S., Blake, M.: Programming human and software-based web services. Computer 43(7), 82–85 (2010)
12. Wei, Y., Blake, M.B.: Service-oriented computing and cloud computing: Challenges and opportunities. IEEE Internet Computing 14(6), 72–75 (2010)

Introduction to the Fifth Workshop on Non-Functional Properties and Service Level Agreements Management in Service-Oriented Computing (NFPSLAM-SOC 2011)

Flavio De Paoli[1], Ioan Toma[2], Carlos Pedrinaci[3], and Marcel Tilly[4]

[1] University of Milano-Bicocca, Italy
depaoli@disco.unimib.it
[2] STI Innsbruck, University of Innsbruck, Austria
ioan.toma@sti2.at
[3] Knowledge Media Institute - The Open University, United Kingdom
c.pedrinaci@open.ac.uk
[4] European Microsoft Innovation Center, Germany
marcel.tilly@microsoft.com

1 Aims and Scope

Nowadays businesses as well as the Web require information to be available in real-time in order to reply to requests, make effective decisions and generally remain competitive. This in turn requires data to be processed in real-time. In general in service-oriented architecture (SOA) one is less concerned with latency in data processing. Clearly, there are investigations of service-level agreements (SLA) and quality of service (QoS) to guarantee service delivery. Research around non-functional properties and service-level agreements for service-oriented computing has reached a level of maturity. There are approaches for describing properties, managing SLAs and even for selecting and composing services based on NFPs. Beyond these classical topics SOA inspired extensions are enabling new and creative domains like the Internet of Things, real-time business or real-time Web. These new domains impose new requirements on SOA, such as a huge data volume, mediation between various data structures and a large number of sources that need to be procured, processed and provided. Questions like how to pick the right service out of tens of thousands of services if we talk about sensor networks or how to provide results with almost near zero-latency describe actual questions and challenges we are currently facing. Therefore, we have to look into new ways for processing data, converting and composing data coming from various sources and for enabling an easy and lightweight way to impose it on various sets of devices.

This is further motivated by the recent evolution of the Web, which is changing radically the way data and services are shared, used, and combined. On the one hand, impelled by Web 2.0 technologies, services on the Web are increasingly shared openly based on REST principles and a light technology stack based largely on HTTP and XML or JSON. On the other hand, there is an ever

G. Pallis et al. (Eds.): ICSOC 2011, LNCS 7221, pp. 97–99, 2012.

increasing amount of data available on the Web provided by RESTful services, social networks, open data initiatives, and even sensors. To cater for this increase in the quantity of data but also in the diversification of its nature, Linked Data principles are emerging as the best means for sharing data on the Web by exposing and interlinking information about any entity on the basis of URIs and RDF. These technologies are giving birth to a Web of Data about basically anything. Driven by the aforementioned trends, the emergence of a Web of Services capturing data about services and their relationship with other services and real-world entities such as people, organizations or products is gradually taking place, see for examples initiatives like GoodRelations and iServe. Finding, composing, invoking and enacting services effectively in this new context is thus increasingly becoming a matter of efficiently and effectively analyzing and combining large amounts of data to make informed and adaptive decisions.

The first four editions of the NFPSLA-SOC Workshop (nfpsla-soc07, second edition, nfpsla-soc09, nfpslam-soc10) were organized at the ICSOC 2007, ECOWS 2008, ICSOC 2009 and ECOWS 2010. They were focused on the management of Non-Functional Properties and Service Level Agreements in the context of Service Oriented Computing. While the general objectives of the workshop remain the same, for the current edition of the workshop we aim to create a forum where one can expose and discuss novel ideas on the use and management of non-functional properties and Service Level Agreements bearing in mind the aforementioned evolution in service technologies and related activities influencing the service world such as social networks, open data initiatives, and even sensors. The workshop aims to tackle the research problems around methods, concepts, models, languages and approaches for management, including finding, composing, invoking and enacting services informed by non-functional properties as well as by any other information related to services. This proposed workshop aims to bring together researchers and industry attendees addressing these issues, to promote and foster a greater understanding of how the management of NFP, QoS and SLAs can assist business to business and enterprise application integration. We are especially interested in new ways utilizing linked data, Web2.0 approaches to enable scalable service related tasks based on non-functional properties and SLA descriptions.

2 Contributions

The workshop was organized in three sections: a keynote and two sections with paper presentations and discussion arranged into two broad themes: "Quality-based service selection", and "Adaptive composition in dynamic environments". The workshop keynote was given by Darko Anicic, whose talk, "Intelligent Complex Event Processing: Opportunities and Challenges" provided a novel perspective for much of the work presented in the workshop papers. The talk focused on the core role of real-time data processing and its capability to support continual monitoring in high dynamic environments and facing data ambiguities. Reasoning about events and their context, and compact (several levels of abstraction) and contextualized way to present information were discussed as key elements to

support an efficient decision making process. The talk has generated interesting discussions on what are the requirements for the next generation of technological platforms and the crucial role of non-functional properties.

In the "Quality-based service selection" section, two papers were presented. The paper "Applying QoS-aware Service Selection on Functionally Diverse Services" proposes a clustering method to leverage background knowledge on the compatibility of the services, and enable heuristic algorithms to discover valid workflow configurations in shorter time. The approach is to integrate the proposed method into a genetic algorithm by performing repair operations on invalid genomes. The solution is compared with related heuristic algorithms that use the same guided target function but pick services in a random manner. The paper "Semantic matching of WS-SecurityPolicy assertions" proposes the transformation of WS-SecurityPolicy (WS-SP) into an OWL-DL ontology and the definition of a set of semantic relations between the provider and requestor security concepts to make the matching of security assertions effective. A discussion on how these relations lead to more correct and more flexible matching of security assertions is also provided.

Issues on "Adaptive composition in dynamic environments" have been discussed in two papers. The first one, "Quality Prediction in Service Composition Frameworks" proposes a novel service composition process that includes QoS prediction for composed services as an integral part. With this approach, systematic consideration of service quality during the composition process is naturally achieved, without the need for detailed knowledge about the underlying prediction models. The QoS prediction support was integrated in a large-scale SLA management framework and a service mashup platform to show how the process can be used in practice. The paper "ECMAF: An Event-Based Cross-Layer Service Monitoring and Adaptation Framework" addresses cross-layer issues in the context of monitoring and adaptation of Service-Based Applications (SBAs) by proposing a framework that adopts a series of techniques, such as event-pattern detection, event monitoring and logging, to support the definition of mappings between event patterns and appropriate adaptation strategies. A case study on Traffic Management is discussed to illustrate the proposed solution.

Applying QoS-Aware Service Selection on Functionally Diverse Services

Florian Wagner[1,⋆], Fuyuki Ishikawa[2], and Shinichi Honiden[1,2]

[1] The University of Tokyo, Japan
[2] National Institute of Informatics, Tokyo, Japan
{florian,f-ishikawa,honiden}@nii.ac.jp

Abstract. The Service-Oriented Computing (SOC) paradigm envisions the composition of loosely coupled services to build complex applications. Most current selection algorithms assume that all services assigned to a certain task provide exactly the same functionality.

However, in realistic settings larger groups of services exist that share the same purpose, yet provide a slightly different interface. Incorporating these services increases the number of potential solutions, but also includes functional invalid configurations, resulting in a sparse solution space. As a consequence, applying naïve heuristic algorithms leads to poor results by reason of the increased probability of local optima.

For that purpose, we propose a functionality clustering in order to leverage background knowledge on the compatibility of the services. This enables heuristic algorithms to discover valid workflow configurations in shorter time. We integrate our approach into a genetic algorithm by performing repair operations on invalid genomes. In the evaluation we compare our approach with related heuristic algorithms that use the same guided target function but pick services in a random manner.

1 Introduction

1.1 Service Composition

Services are interoperable and reusable software components that are published on company intranets and recently more and more on public servers and cloud systems. Standardized interface description languages such as WSDL specify the functionality of these services. Furthermore, services may provide Service-Level Agreements (SLA) that declare the non-functional parameters [8].

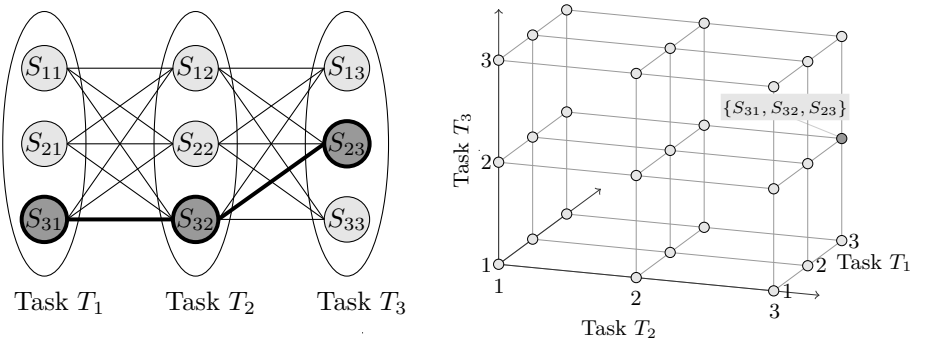
Service composition algorithms are applied in order to provide a virtually unlimited number of functionalities. In general, in the first step a workflow template containing service tasks is provided by the user or by a planning algorithm [12]. A service task is associated with a set of functionally equivalent services, each having varying QoS-attributes. In the next step, a QoS-aware service selection algorithm computes a workflow configuration that determines for each task one single service. The goal is to optimize the utility of the service composition and

⋆ The work of Florian Wagner is partially supported by the KDDI Corporation.

G. Pallis et al. (Eds.): ICSOC 2011, LNCS 7221, pp. 100–113, 2012.

meet the QoS-constraints. Service selection optimizes the consumption of resources and is therefore a major concern for companies, but is also considered as an important issue in research [9].

Since the problem has been proven to be NP-hard [10], enumerating all possible workflow configurations is not feasible. Therefore, approaches that employ near-optimal heuristic algorithms such as hill-climbing [4] and genetic algorithms [1,14] have been applied. All these approaches assume that for each task a set of functionally equivalent services exists. Fig. 1 depicts the service selection problem. In Fig. 1b the solution space is visualized where each solid point is a valid workflow configuration.



(a) All possible service combinations, highlighting one sample workflow configuration

(b) Complete solution space

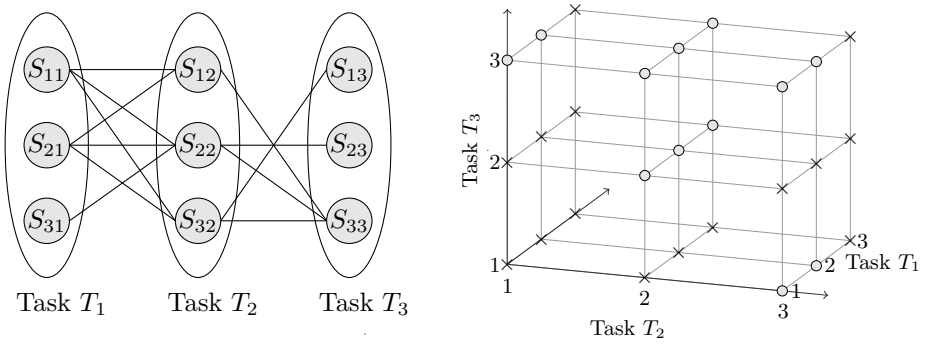
Fig. 1. Classical service selection problem

1.2 Functionally Diverse Services

However, requiring that services associated to a workflow task must have exactly the same functionality might be too restrictive. In reality, various providers offer services that share the same purpose but have slightly varying interfaces, e.g. having optional input parameters or additional result variables. Moreover, service interfaces might be modified and updated during the lifecycle of a service.

Apart from these differences in the interfaces, even if services yield a valid link on the conceptual level, the link may be invalid on the concrete data-structural level [2]. By allowing functionally diverse services in workflow tasks, more candidate services are available, offering more choices to optimize the utility of the workflow.

For these reasons, a certain amount of workflow configurations may be invalid as some services are not functionally compliant. Even though in this scenario the number of possible service combinations in fact decreases, heuristic algorithms achieve only poor results as the solution space is sparse and local optima are more likely. The consequences on the solution space are shown in Fig. 2b. Crosses indicate invalid workflow configurations, e.g. the configuration $\{S_{31}, S_{32}, S_{23}\}$ from Fig. 1a is invalid since there are no valid links between these services.



(a) Restricted service combinations (b) Sparse solution space

Fig. 2. Service selection on functionally diverse services

1.3 Our Contributions

Most related studies present selection algorithms that require functionally equivalent services for each task. Our approach does not have this restriction and is therefore applicable to a wider range of selection problems. Our contributions to the selection problem are as follows:

1. We discuss the implications of introducing functionally diverse services to the selection task. We analyze which heuristic algorithms are applicable to solve the problem and present how these algorithms can be adjusted to find valid workflow configurations and evaluate the resulting performance.
2. We present a method to find valid configurations efficiently based on a functional clustering of candidate services. This method leverages background knowledge on the compliance of the services to prune the search space.
3. We demonstrate how this method can be embedded into a genetic algorithm to repair erroneous genomes. In our detailed evaluation we show which parameters yield the best results for our modified algorithm.

The proposed clustering algorithm is related to the algorithm shown in [11] but uses a more general clustering method. Moreover, the annotations of the graph vary from the previous version and are used for a different purpose.

In the next section we discuss the preliminaries of the selection problem. In Section 3 we present our approach based on a functional clustering of services combined with a genetic algorithm and Section 6 concludes this paper.

2 Preliminaries

In this section we discuss the general service selection problem and further explain the consequences of including functionally diverse services.

2.1 Services

Services provide a certain functionality on a public or private network. In order to specify the functional interface of a service, a description document written in an interface description language such as WSDL is published. Moreover, each service has varying QoS-attributes that are determined in a Service-Level Agreement (SLA) document. In this paper we consider the price, the response time, the reliability, and the availability [13] of a service. A detailed classification on QoS-attributes can be found in [8].

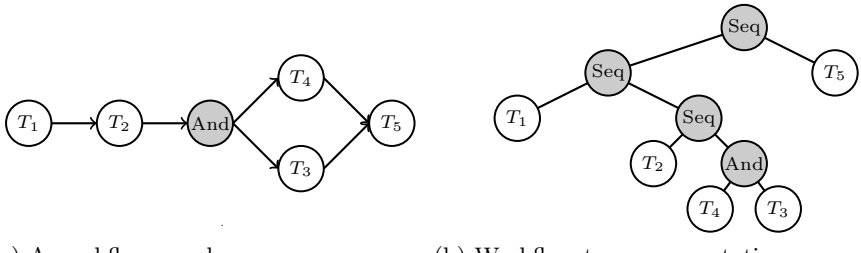
Given the functional interface of a service, service S_1 can be connected to S_2 if S_1 provides an output parameter that is required by S_2.

Comparing a service S_1^T with service S_2^T associated to the same task T, we distinguish three cases:

1. Both services can be connected to the same set of services, in other words the two services are *equivalent*.
2. Service S_1^T can be applied at least in all cases where S_2^T is applicable. In that case we say that S_1^T provides *more functionality* than S_2^T.
3. In all other cases, the two services are *unrelated*.

2.2 Workflows

A workflow is a directed graph where each node is either a task or a control node, depicted in 3a. A task is a set of functional related services that have varying QoS-attributes. A control node determines the control flow of the workflow. In this paper we consider sequences, AND/OR branches, and loops. In the case of loops we "unroll" the loop retrieving the number of expected calls from past invocations. In [3] several possible workflow patterns are discussed in detail.



(a) A workflow graph (b) Workflow tree representation

Fig. 3. Comparison of workflow trees and graph representations. $S_1 - S_5$ depict services, gray nodes are control nodes.

For aggregating the QoS attributes, workflows are often represented as workflow trees, illustrated in 3b. This way, the computations of Table 1 can be applied to the root node. Inner nodes are the control nodes, leaf nodes are the task nodes. For the sake of simplicity we use binary trees in the following.

In order to execute a workflow, a workflow configuration is computed that selects for each task T a service S, where $S \in T$.

QoS-Aggregation. In order to compute the QoS-attributes of a workflow configuration the QoS Q_i of the composite services are aggregated. First, all attributes are normalized between 0 and 1, with 1 being the best value (cf. Equation 1). Therefore, the minimal and the maximum values of the attributes are either pre-defined or determined by comparing all available services. In case the utility of an attribute grows with higher values, the attribute is categorized as a positive attribute Q_{pos}, otherwise as a negative attribute Q_{neg} [6]. For instance, reliability and availability are positive attributes, price and response time are negative attributes.

$$Q'_i = \begin{cases} \frac{Q_i - Q_{min}}{Q_{max} - Q_{min}} & \text{if } Q_i \in Q_{pos} \text{ and } Q_{min} \neq Q_{max}, \\ \frac{Q_{max} - Q_i}{Q_{max} - Q_{min}} & \text{if } Q_i \in Q_{neg} \text{ and } Q_{min} \neq Q_{max}, \\ 1 & \text{else} \end{cases} \tag{1}$$

Subsequently, the normalized attributes Q'_i of the services are aggregated to the QoS $\mathcal{Q}$ of the workflow depending on the characteristics of the attributes and the control flow. Table 1, taken from [1], depicts the aggregation for common control structures and QoS-attributes. The probability p refers to the execution probability of the branch, k is the expected loop count. Both variables are calculated based on either execution logs or estimated by humans. We apply the same computations, but instead of using a probabilistic approach for concurrent execution branches we adopt a worst-case computation.

These computations are first applied to the root node of the workflow tree, iterating subsequently to the child nodes until all nodes have been visited. In each node the QoS of the subtree are aggregated, with the root node containing the aggregated QoS of the workflow.

Table 1. QoS-aggregation of composite services, adopted from [1, Table 1]

Attribute	Sequence	AND-Branch	OR-Branch	Loop
Response time (t)	$\sum_{i=1}^{n} S_i.Q_t$	$\max_{S \in N}(S_i.Q_t)$	$\max_{S \in N}(S_i.Q_t)$	$k \cdot S_i.Q_t$
Price (c)	$\sum_{i=1}^{n} S_i.Q_c$	$\sum_{i=1}^{n} S_i.Q_c$	$\max_{S \in N}(S_i.Q_c)$	$k \cdot S_i.Q_c$
Reliability (r)	$\prod_{i=1}^{n} S_i.Q_r$	$\prod_{i=1}^{n} S_i.Q_r$	$\min_{S \in N}(S_i.Q_r)$	$S_i.Q_r^k$
Availability (a)	$\prod_{i=1}^{n} S_i.Q_a$	$\prod_{i=1}^{n} S_i.Q_a$	$\min_{S \in N}(S_i.Q_a)$	$S_i.Q_a^k$

Utility Computation. After computing the QoS-attributes of the entire workflow, the QoS vector $\mathcal{Q}$ is aggregated to a single value. For that purpose, utility functions are defined, in most studies a Simple Additative Weighting (SAW) is applied as in Equation 2.

$$\mu(\mathcal{Q}) = \sum_{i=1}^{|\mathcal{Q}|} w_i \cdot \mathcal{Q}_i \qquad \text{where} \sum_{i=1}^{|\mathcal{Q}|} w_i = 1 \tag{2}$$

The user provides for each QoS-attribute a certain weight w_i, indicating the user's preferences towards e.g. the price or the response time.

Global QoS-Constraints. Apart from the utility, the user may state global QoS-constraints, determining limits on e.g. the price of the whole workflow composition. In case the constraints are violated, the utility of a workflow becomes 0. Therefore, in this setting QoS-constraints are considered as hard constraints.

2.3 Heuristic Approaches

Since the selection problem is a NP-hard problem, near-optimal heuristic approaches are applied. In case all services provide the same functionality, the functional compliance is always satisfied. Therefore, the algorithms only have to take the QoS-attributes into consideration, by using a heuristic function that subtracts a penalty from the utility if constraints are violated (cf. Eq. 3).

$$\mu_{heu}(\mathcal{Q}) = \mu(\mathcal{Q}) - (w_{qos} \cdot d_{qos}) \tag{3}$$

The value d_{qos} equals the number of violated QoS-constraints multiplied by a certain weight w_{qos}. This way, the algorithm is guided towards a solution that meets the QoS-constraints. In the following, we will discuss two popular heuristic approaches that employ the modified utility function μ_{heu}.

Hill-Climbing. Hill-climbing is a simple greedy approach which tries to improve a selection by replacing a single service in each iteration. First, a random configuration is computed. Next, for a random task the service that yields the best heuristic value is selected.

An extension of the classical hill-climbing algorithm iterates over a set of candidate solutions and in the end chooses the selection with the best heuristic value. This way, the effect of local optima can be reduced but the computation time increases as well.

Genetic Algorithms. Genetic Algorithms maintain a set of genomes that encode a service selection. The genome length equals the workflow length and the cells contain the indices of the selected services, depicted in Fig. 4.

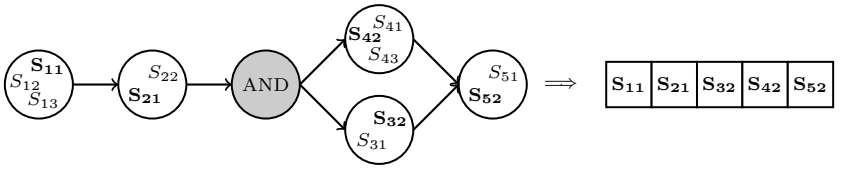


Fig. 4. Translating a workflow configuration (left) to a genome (right)

Genomes are evolved in each generation by applying the following operations:

Crossover given a probability P_{cro}, two genomes are combined by picking a service from the genomes for each task. Alternatively, both genomes are cut into halves and concatenated.

Mutation given a certain probability P_{mut}, for each task the currently selected service is replaced with a random service from the same task.

Selection. In the last step, the genomes are sorted according to their fitness (the utility function μ_{heu}) and only the best ones are kept.

These three steps are iterated either a fixed number of times or until no change occurred to the fitness of the best genome in a certain number of generations. In each iteration the top-k genomes are kept to ensure that these genomes are not discharged.

2.4 Functional Compliance of Services

In case all services associated to a workflow task provide the same functionality, theoretically every possible configuration is executable. However, in reality certain links between services may be invalid. Workflow configurations that contain at least one invalid link cannot be executed.

In most cases, these restrictions can be automatically derived by verifying the type hierarchy of the parameters or the WSDL interface. If a service provides exactly the required output parameters or parameters with additional data to another service, then these services yield a valid link. Apart from that, the user may also declare certain service combinations as invalid, e.g. if two services are hosted on servers that are connected with a low-bandwidth cable. In the following we assume that a relation $combinable : S \times S$ exists, that indicates whether two services can be combined with each other.

Similar to the QoS-constraints, these restrictions are hard constraints, therefore if a selection contains an invalid link, the utility becomes 0.

3 Approach

In this section we present our algorithm that integrates a repair operation for invalid workflow configurations into a genetic algorithm. This operation leverages a functional clustering on the candidate services to resolve invalid links efficiently.

3.1 Functional Clustering

For each workflow task the services are clustered and arranged in a direct-acyclic graph based on their functionality. The functionality of a service S depends on the services that can be connected to S.

Converting the Workflow Tree. Since the functionality of a certain service in a workflow depends on the incoming and outgoing links in a workflow, the workflow tree is first converted into a graph representation to detect these links, described in Algorithm 1. The algorithm starts with the root node, descending first to the task nodes and adding them to the graph G (line 1).

Subsequently, the inner nodes are traversed. In case a sequence is found the incoming task of the left subtree and the outgoing tasks of the right subtree are stored in the sets M_1 and M_2 (line 6 and 7). Then, in line 10 the services of both sets are connected with a directed edge and added to $G.E'$.

Algorithm 1. convert(T, n)

```
   Input: Workflow tree T = (V, E), node pointer n
   Output: Workflow graph G = (V', E')
 1 if n ∈ Tasks then G.V' := G.V' ∪ {n};
 2 else
 3 |   convert(T, n.left);
 4 |   convert(T, n.right);
 5 |   if n is Sequence then
 6 |   |   M1 := outgoingTasks(n.left);
 7 |   |   M2 := incomingTasks(n.right);
 8 |   |   foreach Task t1 ∈ M1 do
 9 |   |   |   foreach Task t2 ∈ M2 do
10 |   |   |   |   G.E' := G.E' ∪ {t1, t2}
11 |   |   |   end
12 |   |   end
13 |   end
14 end
```

In Alg. 2 the helper function `incomingTasks` is shown, computing all services that provide outgoing links in a subtree. In case the parameter n is a task node then the task itself is returned (line 2). Otherwise, if the control node is a sequence then we descend the tree to the right child node (line 4). In case of a parallel control node, the union of the left and right node is returned in line 5.

Algorithm 2. incomingTasks

```
   Input: Node n
   Output: Set M of outgoing tasks in the subtree of n
1 if n ∈ Tasks then
2 |   return {n.t};
3 else
4 |   if n is Sequence then return incomingTasks(n.left);
5 |   else return incomingTasks(n.left) ∪ incomingTasks(n.right);
6 end
```

The helper function `outgoingTasks` is defined in the same way, except for providing `n.right` as a parameter in line 5.

After computing the incoming and outgoing links we determine for each service the set of functional compatible predecessor IL and successor OL services, in other words services that can provide outputs and receive inputs from a service respectively. Given two tasks t and t' where $(t, t') \in G.E'$, for all services

pairs $\{S, S' | S \in t, S' \in t'\}$ that are contained in *combinable*, service S' is added to the set $S'.IL$ and S is added to the set $S'.OL$.

Computing the Functionality Graph. Based on the two sets IL and OL of each service a functionality graph is computed by comparing all services in a task with another. Two rules apply:

1. If $S.IL = S'.IL$ and $S.OL = S'.OL$ then the services are functionally equivalent. In that case, these services are clustered in the same composite node.
2. Otherwise, if $S.IL \subseteq S'.IL$ and $S.OL \subseteq S'.OL$ then S and S' are connected with a directed edge.

Each connected component forms a cluster. In Fig. 5 an example cluster with 12 services is shown, taken from the clustering of the OWLS-TC test[1].

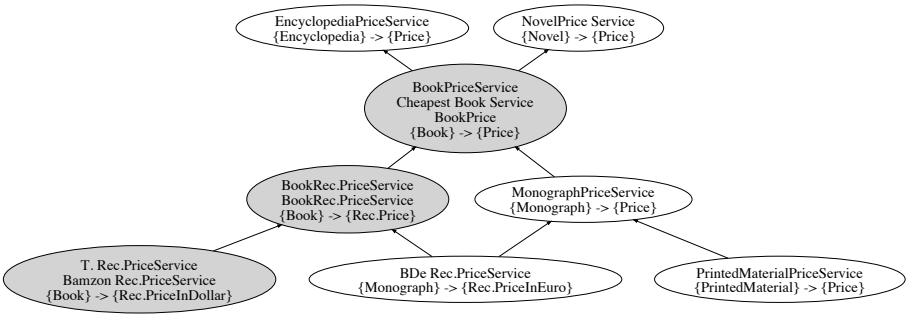
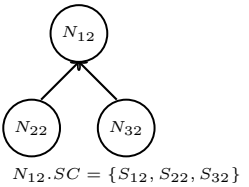


Fig. 5. An example cluster from the resulting clustering of the OWLS-TC testset. Composite services are marked by gray nodes. The relation *combinable* is computed by comparing the parameter types of the services (last line of the nodes).

In the last step, we aggregate in each node the accumulated output list $\mathcal{OL}$ of all OL sets in the subcluster and $\mathcal{SC}$ containing the set of all services in the subcluster.

Nodes	Services	N.IL	N.OL
N_{12}	S_{12}	$\{S_{11}, S_{21}\}$	$\{S_{33}\}$
N_{22}	S_{22}	$\{S_{11}, S_{21}, S_{31}\}$	$\{S_{23}, S_{33}\}$
N_{32}	S_{32}	$\{S_{11}, S_{21}\}$	$\{S_{13}, S_{33}\}$

(a) The computed N.IL and N.OL sets



(b) Resulting graph

Fig. 6. Results of applying the algorithm to Task T_2 of Fig. 2a

[1] http://projects.semwebcentral.org/projects/owls-tc/

The resulting functionality graph of the second task of Fig. 2a is shown in Fig. 6b as an example.

A specific form of the clustering technique is described in [12] where the method is applied to automatic service composition. The functional comparison is based on the parameter types and pre- and postconditions of the services. We present a more general approach to cluster services in this paper that can be applied to any kind of services, even if no type hierarchy and pre- and postconditions are given.

3.2 Combining Functional Clustering with a GA

In our approach we apply a genetic algorithm that performs an intermediate repair operation on the genomes between the mutation and the selection step. The fitness function extends the function μ_{heu} (cf. Eq. 3) in a way that functional compliance is taken into account, shown in Eq. 4.

$$\mu'_{heu}(\mathcal{Q}) = \mu(\mathcal{Q}) - (w_{qos} \cdot d_{qos} + w_{func} \cdot d_{func}) \tag{4}$$

The variable d_{func} indicates the ratio of invalid service links and w_{func} assigns a certain weight to this value. This function is used in the genetic algorithm as a fitness function of the genomes to guide the algorithm towards selections that are functional compatible and meet the QoS constraints.

In each generation an invalid genome is repaired with a certain probability P_{rep}. In general, increasing the repair probability performs repair operations more aggressively but also increases the computational complexity of each generation. By experiment we have evaluated that 33% is a good tradeoff between utility and performance.

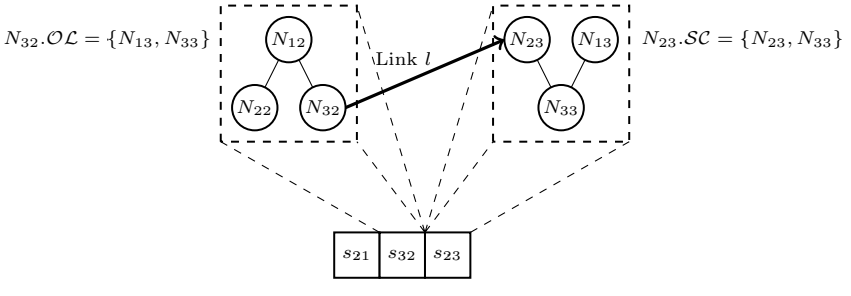


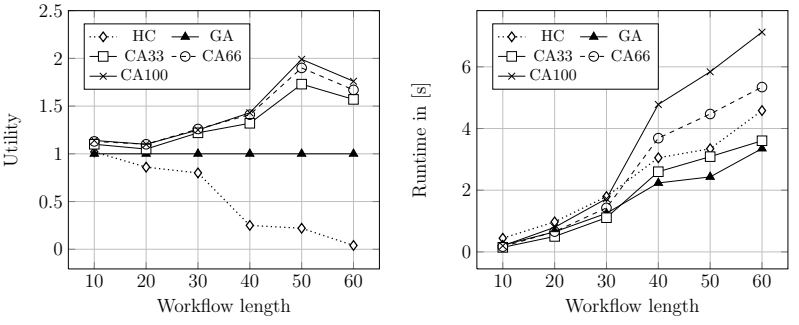
Fig. 7. Repairing an invalid link of a genome

In order to repair an invalid genome, services S_{in} and S_{out} that are part of an invalid link are detected and replaced. First, we detect whether the invalid link can be resolved by replacing one service with a service from its subtree. This is the case when the intersection of $S_{in}.\mathcal{OL}$ and $S_{out}.\mathcal{SC}$ is not empty, depicted in Fig. 7. In the example, N_{23} is replaced by N_{33}.

In case the intersection is empty, one of the parent nodes of S_{in} and S_{out} is selected randomly. If both nodes are already root nodes, then we pick a random root node from a different cluster of the same task.

4 Evaluation

In our evaluations, we use a hill-climbing algorithm (HC) that iterates 100 initial solutions 100 times, therefore 10.000 iterations in total. The genetic algorithm (GA) has 200 seeds, iterating at most 200 times. Furthermore, we evaluate our cluster algorithm (CA) that is based on the GA, including the repair operation and with 100 seeds. We provided various repair probabilities in order to evaluate the optimal parameter, e.g. CA33 is a CA with 33% repair probability. All experiments regarding the performance have been executed on an Intel Quad-core 3.00Ghz with 4GB ram.



(a) Utility comparison (GA baseline) (b) Comparing the runtime

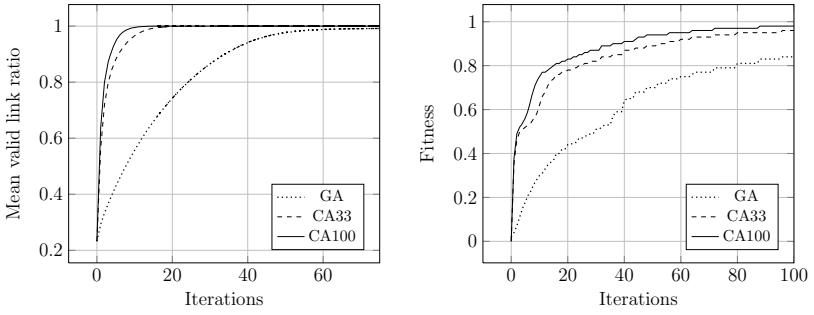
Fig. 8. Evaluating the utility and performance with varying workflow length

We have created workflows with varying lengths, consisting of sequences and AND/OR branches. Each task contains 20 concrete services. The QoS attributes of the services were chosen randomly, except for the price which was chosen partially randomly and partially depending on the other attributes.

In the first two experiments, each test case is executed 1,000 times with varying QoS constraints on price and response time. We set the utility of the GA as the baseline to 100% in order to evaluate the benefit of the additional repair operation. In Fig. 8a and 8b we vary the workflow length with a fixed average service compatibility of approx. 40%.

In relation to the baseline, the utility of the HC algorithm decreases with growing workflow length, whereas the utility of the CA grows (cf. Fig. 8a). Regarding the performance, with increasing repair probability more runtime is required as shown in Fig. 8b. Based on these results, we choose the CA33 algorithm as a reasonable tradeoff between utility and performance.

In the last experiments we evaluate the evolution of the genomes. We use a workflow of length 60 with service compliance 20%. In Fig. 9a we compare the mean ratio of valid links of all genomes. The genetic algorithm repairs all genomes in approx. 50 iterations. In contrast, the CA repairs the genomes in 10 (CA33) and 5 (CA100) iterations respectively. In Fig. 9b the fitness value of the best genomes are compared. For instance, the fitness of the GA in iteration 60 is achieved by the CA in iteration 10.



(a) Ratio of valid links of all genomes (b) Convergence of the fitness

Fig. 9. Comparing the evolution of the genomes

5 Related Work

We review some representative selection approaches that deal with the selection problem for functionally equivalent services and studies that analyze the functional compliance of services.

Zeng et al. formulate in [13] the composition problem as a Multi-Knapsack Decision Problem, proposing Integer Programing (IP) to tackle the problem. Their approach computes an optimal solution, but with growing problem size this approach cannot compute a solution in feasible time. Moreover, invalid workflow configurations are difficult to integrate in this approach.

In order to overcome the performance issues by computing an optimal solution with IP, near-optimal heuristic approaches such as hill-climbing [4] and genetic algorithms [1] have been investigated. A penalty is subtracted from the utility if constraints are violated in order to guide the algorithm to a feasible solution.

Nebil et al. [6] apply a clustering method on the service tasks before the actual selection process. In contrast to our approach they apply the clustering on the QoS-attributes of the services. For that purpose, they employ a k-nearest neighbor algorithm to identify the clusters. The algorithm computes a solution in reasonable time but the achieved utility is rather low $(60\% - 80\%)$.

Similar to our approach, Lécué and Mehandjiev apply in [5] a genetic algorithm to optimize the QoS and the matching quality of the service links. For that purpose, they combine these two values in the fitness function of the genomes. As a consequence, configurations with better QoS attributes but less functional compliance might be ranked higher. Contrary to their approach we treat the functional compliance like a QoS constraint, adding a penalty factor to the fitness of invalid genomes.

In [2] the authors investigate the gap between the conceptual and data-structural level of service composition. They claim that even if the conceptual types of the parameters of two services are compatible, the WSDL implementation may be incompatible. The focus of their work is on the automatic composition, taking no QoS-attributes into account.

The authors of the EASY project [7] arrange services in a functionality graph. The edges of this graph are based on the different Plug-in relationship as we apply it. QoS attributes are not considered in this work.

6 Conclusion and Future Work

In this paper we have discussed the problem of applying QoS-aware service selection on functionally diverse services and shown the impact on heuristic algorithms. We have presented which adjustments are necessary to apply heuristic algorithms and evaluated the algorithms on the modified problem. In the next step we have integrated a repair operation for the erroneous genomes based on a functional clustering into a genetic algorithm. This modification leverages background knowledge on the service compliance. We have shown that this modification outperforms naïve heuristic algorithms that randomly select services without taking the service compliance into account. Moreover, we have determined the optimal parameters for our algorithm by experiment.

As a next step we intend to improve the integration of the functional compliance criteria that is currently based on adding a penalty to the target function, similar to QoS-constraints. We will consider multi-objective optimization to balance the conflicting functional and non-functional requirements.

Furthermore, we plan to investigate on how to integrate the proposed solution into our automatic service composition approach [12]. The approach computes workflow templates that contain tasks with functionally diverse services.

References

1. Canfora, G., Di Penta, M., Esposito, R., Villani, M.L.: An approach for QoS-aware service composition based on genetic algorithms. In: Proceedings of the Conference on Genetic and Evolutionary Computation, GECCO (2005)
2. Ishikawa, F., Katafuchi, S., Wagner, F., Fukazawa, Y., Honiden, S.: Bridging the Gap Between Semantic Web Service Composition and Common Implementation Architectures. In: IEEE International Conference on Services Computing, SCC (2011)
3. Jaeger, M.C., Rojec-Goldmann, G., Mühl, G.: QoS Aggregation for Web Service Composition using Workflow Patterns. In: EDOC (2004)
4. Klein, A., Ishikawa, F., Honiden, S.: Efficient Heuristic Approach with Improved Time Complexity for QoS-aware Service Composition. In: IEEE International Conference on Web Services (ICWS), Washington D.C., USA (2011) (to appear)
5. Lécué, F., Mehandjiev, N.: Seeking Quality of Web Service Composition in a Semantic Dimension. IEEE Trans. on Knowl. and Data Eng. 23, 942–959 (2011)
6. Ben Mabrouk, N., Beauche, S., Kuznetsova, E., Georgantas, N., Issarny, V.: QoS-Aware Service Composition in Dynamic Service Oriented Environments. In: Bacon, J.M., Cooper, B.F. (eds.) Middleware 2009. LNCS, vol. 5896, pp. 123–142. Springer, Heidelberg (2009)
7. Mokhtar, S.B., Preuveneers, D., Georgantas, N., Issarny, V., Berbers, Y.: EASY: Efficient semantic service discovery in pervasive computing environments with QoS and context support. Journal of Systems and Software 81(5), 785–808 (2008)

8. O'Sullivan, J., Edmond, D., ter Hofstede, A.H.M.: What's in a Service? Distributed and Parallel Databases 12(2/3), 117–133 (2002)
9. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F.: Service-Oriented Computing: State of the Art and Research Challenges. IEEE Computer 40(11) (2007)
10. Pisinger, D.: Algorithms for Knapsack Problems. Ph.D. thesis, DIKU, University of Copenhagen, Denmark, Technical Report 95-1 (1995)
11. Wagner, F.: Efficient, Failure-Resilient Semantic Web Service Planning. In: Maglio, P.P., Weske, M., Yang, J., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6470, pp. 686–689. Springer, Heidelberg (2010)
12. Wagner, F., Ishikawa, F., Honiden, S.: Qos-aware automatic service composition by applying functional clustering. IEEE International Conference on Web Services, ICWS (2011)
13. Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., Sheng, Q.Z.: Quality Driven Web Services Composition. In: Proceedings of the 12th International Conference on World Wide Web (WWW), pp. 411–421. ACM (2003)
14. Zhang, W., Schütte, J., Ingstrup, M., Hansen, K.M.: A Genetic Algorithms-Based Approach for Optimized Self-protection in a Pervasive Service Middleware. In: Baresi, L., Chi, C.-H., Suzuki, J. (eds.) ICSOC-ServiceWave 2009. LNCS, vol. 5900, pp. 404–419. Springer, Heidelberg (2009)

Semantic Matching of WS-SecurityPolicy Assertions

Monia Ben Brahim, Tarak Chaari, Maher Ben Jemaa, and Mohamed Jmaiel

ReDCAD Laboratory, University of Sfax, National School of Engineers of Sfax,
BP 1173, 3038 Sfax, Tunisia
monia.benbrahim@gmail.com, tarak.chaari@redcad.org,
{maher.benjemaa,mohamed.jmaiel}@enis.rnu.tn
http://www.redcad.org

Abstract. The lack of semantics in WS-SecurityPolicy (WS-SP) hampers the effectiveness of matching the compatibility between WS-SP assertions. To resolve this problem, we present in this paper a semantic approach for specifying and matching the security assertions. The approach consists in the transformation of WS-SP into an OWL-DL ontology and the definition of a set of semantic relations that can exist between the provider and requestor security concepts. We show how these relations lead to more correct and flexible matching of security assertions.

1 Introduction

Dynamic service discovery and selection is an essential aspect of Service Oriented Architecture (SOA). To meet modern business requirements, service selection must not only take into account functional aspects, but also non-functional properties of the service. This paper focuses on message security which is one of these non-functional properties. Message security becomes a major concern when using Web services, the most adopted implementation of SOA. Message security mainly means the confidentiality and the integrity of data transmitted through the message. Confidentiality and integrity can be assured by applying security mechanisms such as encryption and digital signature.

WS-Security [1] and WS-SecurityPolicy (WS-SP) [2] are the most important standards for definition and enforcement of message security in systems based on Web services. While WS-Security defines the syntax of security elements in a SOAP message, a WS-SP document describes which security measures are to be applied to which parts of the message. The basic building blocks of SPs are security assertions that consider the WS-Security message security model. Every single assertion may represent a specific requirement, capability, other property, or a behavior related to message security [2]. WS-SP is built on top of WS-Policy framework [3].With the help of WS-Policy operators, security assertions can be composed to complex security policies.

WS-SP is widely accepted in the industry and it is currently a popular standard to be aggregated into the Web service architecture. Nevertheless, WS-SP has a major weakness: it only allows syntactic matching of security policies. In

G. Pallis et al. (Eds.): ICSOC 2011, LNCS 7221, pp. 114–130, 2012.

fact, security policy matching depends on the policy intersection mechanism [3] provided by WS-Policy. The core step in this mechanism is matching the assertions specified in the service provider and requestor policies. This step consists in a pure syntactic comparison between the security assertions, neglecting the domain-specific semantics of assertions such as message security semantics. Although WS-Policy admits that checking the compatibility of policy assertions may involve domain-specific processing, it does not give hints on how to integrate it. Syntactic matching of security assertions restricts the effectiveness of checking the compatibility between them. In fact a simple comparison of the syntactic descriptions of security assertions is prone to get fault negative results. For example, consider syntactically different security assertions with the same meaning. Such assertions are considered incompatible which is counterintuitive. Besides, syntactic matching of security assertions only yields a strict yes/no matching result. A more flexible matching with intermediary matching degrees is needed in order to consider subtle differences that may exist between security assertions and not reject a potential partnership between a service requestor and provider. For example consider the cases when the provider security assertion and the requestor security assertion have the same type but have some different properties that make the provider assertion stronger, from security perspective, than the requestor assertion. Such assertions are considered incompatible which is overly strict.

In this paper we propose an approach to enable semantic matching of Web service security assertions. Firstly, we transform WS-SP into an ontology. Secondly, we extend this WS-SP-based ontology with new semantic relations, such as isEquivalentTo and isStrongerThan relations, between the provider and requestor security assertions. The additional relations allow to semantically interpret the syntactic heterogeneities that may exist between these assertions, particularly when the provider and the requestor security assertions point to the same ontological concept but have different properties. We define a set of SWRL [4] based semantic rules in order to control the dynamic instantiation of the additional semantic relations. Then, we propose an algorithm for matching security assertions.

We implemented a prototype for the semantic matching of security assertions. We used this prototype to match several syntactically different but semantically related assertions and obtained the expected matching degrees between them. Based on the semantic interpretation of the syntactic heterogeneities that may occur between a provider assertion and a requestor assertion, our approach doesn't produce fault negative results and thus supports more correct matching. Besides, it allows to introduce close match and possible match as intermediary matching degrees, which makes security assertion matching more flexible.

The rest of the paper is structured as follows. In Section 2 we present a motivating example to show the need for semantics in matching a pair of requestor-provider assertions. In Section 3 we describe how WS-SP can be extended to incorporate semantics. We firstly present the ontological representation of WS-SP, and then detail the new semantic relations that we propose in order to

semantically interpret the syntactic heterogeneities that may occur between security policy assertions. In Section 4 we present our algorithm for matching security assertions. Section 5 focuses on the implementation of our approach and its application to match examples of assertions. Related work is presented in section 6. Finally, we conclude in Section 7 and give the guidelines of our future work.

2 The Need for Semantics in Matching Security Assertions

Syntactic matching of security assertions restricts the effectiveness of checking the compatibility between them. In order to illustrate this deficiency, suppose that a requestor is looking for a flight reservation Web service that supports the signature of the message body with a symmetric key securely transported using an X509 token. Besides, the necessary cryptographic operations must be performed using Basic256 algorithm suite. This could be formalized, based on the WS-SP standard, by adding the assertions *RAss1* and *RAss2* from Fig. 1 to the requestor security policy (SP). Furthermore, suppose that the requestor finds a Web service that provides flight reservation and whose SP includes *PAss1* and *PAss2* assertions (see Fig. 1). In order to know if the requestor and provider SPs are compatible, we have to check the compatibility of their assertions. It is clear that the assertions specified in the provider SP are syntactically different than those specified in the requestor SP. Syntactic matching will then produce a no match result for these assertions. However, semantic interpretation of the above syntactic heterogeneities leads to decide a different matching result. In fact, in the above scenario *RAss1* and *PAss1* assertions have the same meaning: sign the body of the message. Therefore, matching these two assertions must lead to a perfect match rather than to a no match. Besides, the only differ-ence between *RAss2* and *PAss2* assertions is that the `SymmetricBinding` as-sertion specified in the provider SP contains an extra child element which is

RAss1:
```
<sp:SignedEdElements>
   <sp:XPath>
      /Envelope/Body
   </sp:XPath>
</sp:SignedEdElements>
```

RAss2:
```
<sp:SymmetricBinding>
   <sp:ProtectionToken>
      <sp:X509Token/>
   </sp:ProtectionToken>
   <sp:AlgorithmSuite>
      <sp:Basic256/>
   </sp:AlgorithmSuite>
</sp:SymmetricBinding>
```

PAss1:
```
<sp:SignedParts>
   <sp:Body/>
</sp:SignedParts>
```

PAss2:
```
<sp:SymmetricBinding>
   <sp:ProtectionToken>
      <sp:X509Token/>
   </sp:ProtectionToken>
   <sp:AlgorithmSuite>
      <sp:Basic256/>
   </sp:AlgorithmSuite>
   <sp:IncludeTimeStamp/>
</sp:SymmetricBinding>
```

Fig. 1. Example of syntactically different but semantically related security assertions

`sp:IncludeTimestamp`. The meaning of this element is the following: a timestamp element must be included in the security header of the message. From security perspective this strengthens the integrity service ensured by the message signature [16] and thus makes *PAss2* assertion stronger than *RAss2* assertion. Although it is not a perfect match, it is also overly strict to consider that it is a no match case. In fact, if the requestor can strengthen his security assertion by the inclusion of a timestamp, the perfect compatibility between both assertions will be ensured. We consider that it is more flexible to decide a possible match for this case in order to not reject a potential partnership between the service requestor and provider.

We will show in the rest of this paper how the inclusion of semantics into WS-SP enables more correct and flexible matching of security assertions.

3 Extending WS-SecurityPolicy to Incorporate Semantics

The assertions defined in WS-SP must be augmented with semantic information in order to enable semantic matching. Below we describe how we transform WS-SP into an ontology and detail the semantic relations that we add at the level of the security concepts.

3.1 WS-SecurityPolicy-Based Security Policy Ontology

We redesign WS-SP with an ontological representation of its assertions in order to obtain a WS-SP-based ontology that can be augmented with new semantic relations. A graphical depiction of the main parts of the ontology is shown in Figures 2 to 5. The blue coloured object properties represent the additional semantic relations and will be explained in the next subsection.

Web service security assertions are specified within security policies of the service provider and requestor. Typically, the structure of these policies is compliant with the WS-Policy normal form. In the normal form, a policy is a collection of alternatives, and an alternative is a collection of assertions. It is in the assertion components that a policy is specialized. Fig. 2 shows the main classes of the WS-SP-based ontology. As it is illustrated in this figure, we create the three classes *SecurityPolicy*, *SecurityAlternative* and *SecurityAssertion* in order to enable specifying security assertions within security policies. Then, we transform WS-SP assertions into classes and properties in the security policy ontology based on the semantic meaning of these assertions. *SecurityBinding*, *SupportingSecurityTokens*, *TokenReferencingAndTrustOptions*, and *ProtectionScope* are the security assertions of our ontology and are modelled as subclasses of the *SecurityAssertion* class. Due to space limitations, we will focus, in the rest of the paper on *SecurityBinding* and *ProtectionScope* classes.

SecurityBinding class: represents the various security bindings defined in WS-SP (see Fig. 3). This class allows specifying the main security mechanism to apply for securing message exchanges. Security binding can be either transport level, represented by *TransportBinding* class, or message level represented by

MessageSecBinding class that possesses the two subclasses *SymmetricBinding* and *AsymmetricBinding*. Security bindings have common properties of the following types:

- *SecurityHeaderLayout*:security header layouts are represented by instances of this class. *LayoutValue* is a string data type property having a possible value among "Strict", "Lax", "LaxTsFirst", and "LaxTsLast".
- *AlgorithmSuite*:algorithm suites are captured by this class. The possible values of the string property *AlgSuiteValue* are all algorithm suites defined in WS-SP such as Basic256, Basic192, and Basic256Rsa15 algorithm suites.
- *Timestamp*: the presence of a timestamp in the SOAP message is captured by this class having a boolean data type property called *IncludeTimestamp*.

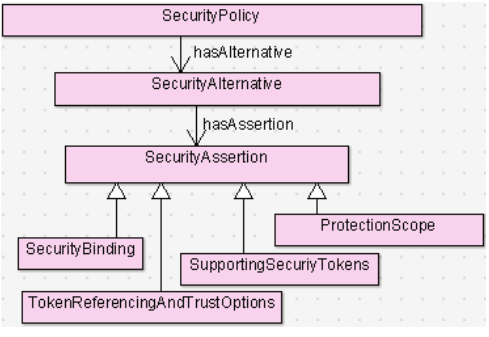


Fig. 2. Security policy ontology: main classes

Message level security bindings have also some common properties such as protection order, signature protection, and token protection that are modelled as data type properties or object properties of the *MessageSecBinding* class.

Besides, each specific type of security binding has binding specific token properties modelled by object properties such as *hasTransportToken*, *hasSignatureToken*, *hasProtectionToken*,etc. These object properties relate a security binding to the *SecurityToken* class (see Fig. 4). This class allows to specify the type of tokens to use to protect or bind claims to the SOAP message. There are different types of tokens with different manners to attach them to messages. Fig. 4 just shows some token types.

ProtectionScope class: is used to group and organize the protection assertions of WS-SP. Fig. 5 shows the property layers of this class. The first level properties are of the following types:

- *EncryptionScope*:this class is used to specify message parts that are to be protected by encryption. Its properties *hasEncryptedPart*, *hasEncryptedElt*, and *hasContentEncryptedElt* allow to specify coarse-grained as well as fine-grained encryption. The *EncryptedPart* class represents a message part to be encrypted. "Body", "Header" and "Attachments" are the possible values of its *PartType* property.

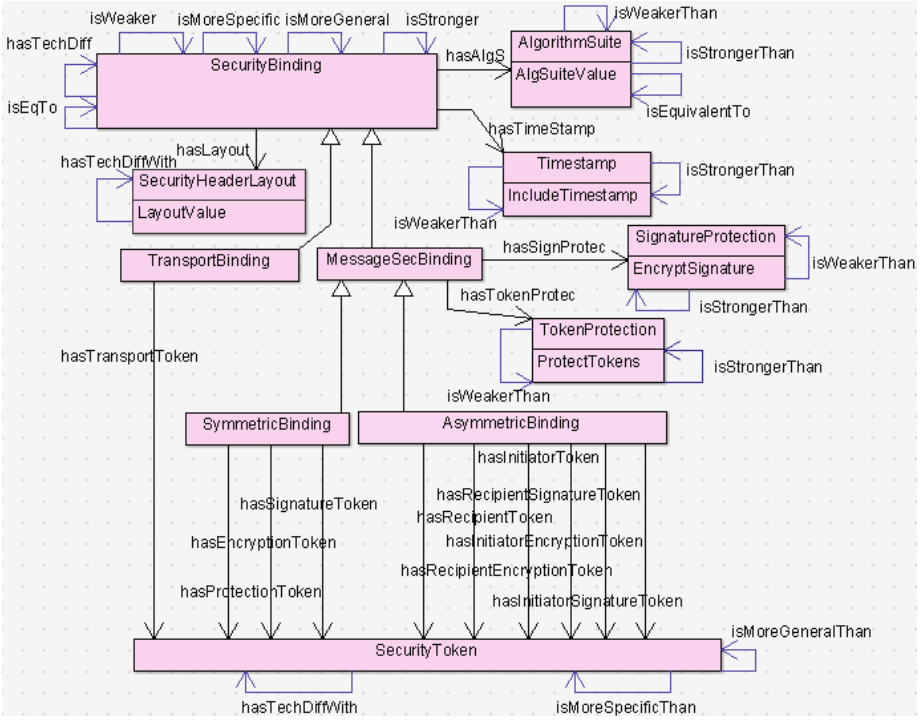


Fig. 3. Security policy ontology: security binding

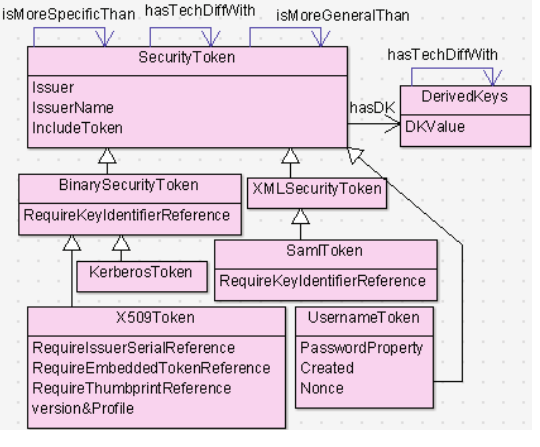


Fig. 4. Security policy ontology: security token

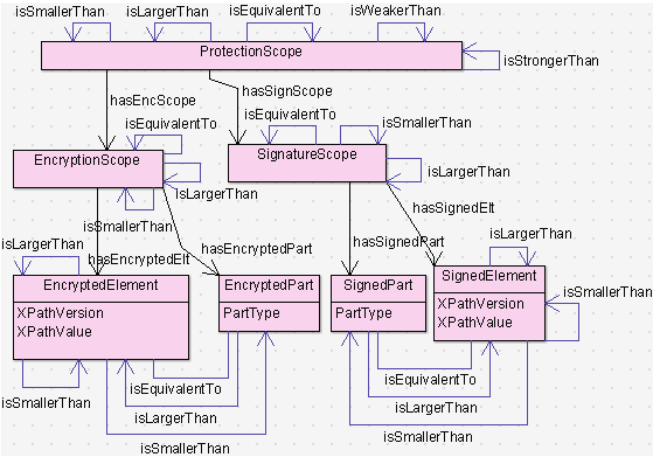


Fig. 5. Security policy ontology: protection scope

– *SignatureScope*:through its properties *hasSignedPart* and *hasSignedElt*, this class allows to describe message parts that are to be digitally signed.
– *RequiredScope*(not shown in the figure due to space limitations): this class is used to specify the set of header elements that a message must contain.

3.2 Adding Semantic Relations

We augment the WS-SP-based ontology with new semantic relations at the level of the security assertions and their properties. These relations are graphically illustrated in Figures 2 to 5 as blue coloured object properties of the ontology classes. They actually link the security concepts of the provider SP to those of the requestor SP.

The automatic instantiation of the semantic relations between concrete provider and requestor security assertions (mapped to instances of the security ontology concepts) is controlled by SWRL [4] based semantic rules. These rules define the conditions that the concrete assertions must satisfy in order to create a given semantic relation between them.

Next, we detail the semantic relations and present (in natural language) some of the semantic rules that control their instantiation.

isIdenticalTo relation[1] . This relation allows to specify that a security concept specified by the provider is identical to a security concept specified by the requestor (no syntactic heterogeneity exists between them). There are two main cases for this relation:

Case 1. It is defined at the level of every security property of ProtectionScope and SecurityBinding assertions. At the level of an atomic property such as

[1] *isIdenticalTo* and *isDifferentFrom* relations are not shown in the figures to avoid saturating them.

Timestamp and AlgorithmSuite, a property specified in the provider assertion is identical to a property specified in the requestor assertion if they point to the same security concept and have equal values. At the level of a composite property such as *SecurityToken* and *EncryptionScope*, a property specified in the provider assertion is identical to a property specified in the requestor assertion if they point to the same security concept and all their child properties are identical (have isIdenticalTo relation).
Case 2. *isIdenticalTo* relation is defined at the level of *ProtectionScope* and *SecurityBinding* assertions. A provider security assertion is identical to a requestor security assertion, if they point to the same security concept and all their properties are identical.

isEquivalentTo relation. This relation allows to specify that a security concept specified by the provider is equivalent to a security concept specified by the requestor. There are three ways this relation can occur:

Case 1. It is defined between an encrypted part and an encrypted element, similarly between a signed part and a signed element, and between a required part and a required element. This is because a message part can also be referenced through an XPath expression. For example, the XPath expression /S:Enveloppe/S:Body references the body part of a SOAP message. The following is one of the semantic rules representing this equivalence case:
If there exists, in the requestor policy, a signed element *reqSignedElt*, which has its XPathValue property having the value "/S:Enveloppe/S:Body", and **if** there exists, in the provider policy, a signed part *PSignedPart*, which has its PartType property, which has the value "Body", **then** create an *isEquivalentTo* relation between *PSignedPart* and *RSignedElt*.
Case 2. *isEquivalentTo* relation is defined between two algorithm suites. An algorithm suite denes which algorithms must be used when cryptographic operations such as signing, encryption, and generating message digests are involved. The algorithm suites dened in WS-SP are not disjoint and two algorithm suites can include many common elements. For example, Basic256 and Basic256Sha256 algorithm suites differ only by the digest algorithm. Therefore, if one of these two algorithm suites is specied in the requestor SP and the other is specied in the provider SP and neither of the two SPs involves digest generation, then the two algorithm suites can be considered as equivalent in this context.
Case 3. This case is a consequence of cases 1 and 2. In fact, if the equivalence relation actually exists at the level of the aforementioned concepts of cases 1 and 2, it spreads to the upper concepts of the security ontology. It can so exist between two encryption scopes, two signature scopes, and two required scopes. Then, it can exist at the level of *SecurityBinding* and *ProtectionScope* assertions. For example, an assertion *PProtecScope* specified by the provider is equivalent to an assertion *RProtecScope* specified by the requestor if it has at least one property (encryption, signature, or required scope) that is equivalent to the corresponding property of RProtecScope, and the other properties of *PProtecScope* and *RProtecScope* are identical.

isLargerThan relation. This relation only concerns the protection scope concept. There are three cases for this relation:

Case 1. Since the body part of a SOAP message is larger than any element that belongs to it, also a header part is larger than any of its sub-elements, we dene *isLargerThan* relation between an encrypted part and an encrypted element, between a signed part and a signed element, as well as between a required part and a required element. Similarly, since any XML element is larger than any of its sub-elements, we furthermore dene isLargerThan relation between two encrypted elements, two signed elements, and between two required elements.
Case 2. *isLargerThan* relation can also exist at the level of encryption scopes, signature scopes, and required scopes. For example an encryption scope *PEncScope* specied by the provider is larger than an encryption scope *REncScope* specied by the requestor if: 1) it has at least an encrypted part or an encrypted element which is larger than an encrypted element of *REncScope*, the other encrypted parts and encrypted elements of *PEncScope* and *REncScope* are identical or equivalent, or 2) it has at least one extra encrypted part or encrypted element than *REncScope*, the other encrypted parts and encrypted elements of *PEncScope* and *REncScope* are identical, equivalent, or have *isLargerThan* relations.
Case 3. At the level of the protection scope concept, a protection scope *PProtecScope* specied by the provider is larger than a protection scope *RProtecScope* specied by the requestor if it has at least one property (encryption, signature, or required scope) that is larger than a property of *RProtecScope*, and the other properties of *PProtecScope* and *RProtecScope* are identical or equivalent. Note that in order to obtain correct *isLargerThan* relations, any syntactic heterogeneity between XML data referenced in the protection scopes must be resolved [10] before executing the involved semantic rules.

isSmallerThan relation. This relation is to specify that the protection scope (in the SOAP message) specified by the provider is smaller than the protection scope specified by the requestor. It is the opposite of *isLargerThan* relation and occurs in three cases just in an opposite manner to *isLargerThan* relation.

isStrongerThan relation. This relation is to specify that a security concept specified by the provider is stronger, in the security perspective, than a security concept specified by the requestor. There are three ways this relation can occur:

Case 1. Since some security binding properties such as algorithm suite, timestamp, signature protection, and token protection have influence on the security strength, we add *isStrongerThan* relation at the level of these properties (see Fig. 3). For instance, the following semantic rule defines when a timestamp property specified in the provider security binding assertion is stronger than a timestamp property specified in the requestor security binding assertion:
If there exists, in the requestor policy, a security binding *RSecB*, and **if** there exists, in the provider policy, a security binding *PSecB*, and if *PSecB* and *RSecB* have the same class of binding (transport, symmetric, or asymmetric), and if *RSecB* has a timestamp *RTs*, which has its *IncludeTimestamp* property which has the value "false", and **if** *PSecB* has a timestamp *PTs*, which has its

IncludeTimestamp property which has the value "true", **then** create an *isStrongerThan* relation between *PTs* and *RTs*.
Case 2. At the level of a security binding assertion, an assertion *PSecB* specified by the provider is stronger than an assertion *RSecB* specified by the requestor if it points to the same type of security binding as *RSecB* but it has at least one property that is stronger than the corresponding property of *RSecB*, and the other properties of *PSecB* and *RSecB* are identical or equivalent.
Case 3. *isStrongerThan* relation is dened between two protection scopes. A protection scope *PProtecScope* specified by the provider is stronger than a protection scope *RProtecScope* specified by the requestor if it has at least one property (encryption, signature, or required scope) that isn't specied in *RProtecScope*, and the other properties of *PProtecScope* and *RProtecScope* are identical, equivalent, or have *isLargerThan* relations.

isWeakerThan relation. This relation is to specify that a security concept specified by the provider is weaker, in the security perspective, than a security concept specified by the requestor. It is the opposite of *isStrongerThan* relation and occurs in three cases just in an opposite manner to *isStrongerThan* relation.

hasTechDiffWith relation. In addition to concepts that allow to specify how a SOAP message is to be secured (confidentiality, integrity,), WS-SP-based ontology also includes concepts to describe technical aspects concerning how adding and referencing the security features in the message. At the level of these technical concepts, we define *hasTechDiffWith* relation to state that any mismatch between the provider concept properties and the requestor concept properties must be considered as a technical mismatch rather than a security level mismatch. There are three cases for this relation:

Case 1. Since the security header layout property simply defines which layout rules to apply when adding security items to the security header and has no influence on the security services ensured by a SP, we added *hasTechDiffWith* relation at the level of the *SecurityHeaderLayout* concept (see Fig. 3). The following semantic rule controls the instantiation of *hasTechDiffWith* relation at the level of the *SecurityHeaderLayout* concept:
If there exists, in the requestor policy, a security binding *RSecB*, and if there exists, in the provider policy, a security binding *PSecB*, and **if** *PSecB* and *RSecB* have the same class of binding, and if *RSecB* has a securityHeaderLayout *RSHL*, which has a LayoutValue property *RHLValue*, and **if** *PSecB* has a SecurityHeaderLayout *PSHL*, which has a LayoutValue property *PHLValue*, and if *RHLValue* and *PHLValue* are not equal, **then** create a *hasTechDiffWith* relation between *PSHL* and *RSHL*.
Case 2. Similarly some child properties of the security token property have no influence on the security services ensured by a SP. So we also add *hasTechDiffWith* relation at the level of these subproperties as well as at the level of the *SecurityToken* concept. For instance, we defined *hasTechDiffWith* relation at the level of *DerivedKeys* property since this property just specifies which mechanism must be applied in order to reference, in the SOAP message, derived keys used

in cryptographic operations and it has no influence on which cryptographic operations to use or with which strength they must be applied.
Case 3. At the level of a security binding assertion, an assertion *PSecB* specified by the provider has technical difference with an assertion *RSecB* specified by the requestor if it points to the same type of security binding as *RSecB* but it has at least one property (*SecurityHeaderLayout* or *SecurityToken*) that has technical difference with the corresponding property of *RSecB*, and the other properties of *PSecB* and *RSecB* are identical or equivalent.

isMoreSpecificThan relation. According to WS-SP standard, many security properties are optional to specify in a SP and WS-SP doesn't attribute default values for them. Therefore we define *isMoreSpecificThan* relation that occurs when a security concept specified by the provider is more specific (i.e., described in more detail) than a security concept specified by the requestor. There are two cases for this relation:

Case 1. It occurs between two *SecurityToken* properties that point to the same concept but the security token specified by the provider is described in more detail than the security token specified by the requestor. For example, the security token specified by the provider is an X509 token that has its property *Version&Profile* having the value "WssX509V3Token10" (which means that an X509 Version 3 token should be used as defined in WSS:X509TokenProfile1.0 [5]), while the security token specified by the requestor is an X509 token with no properties.
Case 2. *isMoreSpecificThan* relation is defined at the level of a security binding assertion, an assertion *PSecB* specified by the provider is more specific than an assertion *RSecB* specified by the requestor if: 1) it points to the same type of security binding as *RSecB* but it has a security token property that is more specific than the corresponding security token property of *RSecB*, and the other properties of *PSecB* and *RSecB* are identical or equivalent, or 2) it points to the same type of security binding as *RSecB* and has a *SecurityHeaderLayout* property (which is an optional property without a default value), while *RSecB* hasn't a *SecurityHeaderLayout* property, and the other properties of *PSecB* and *RSecB* are identical or equivalent.

isMoreGeneralThan relation. This relation occurs when a security concept specified by the provider is more general (i.e., described in less detail) than a security concept specified by the requestor. It is the opposite of isMoreSpecificThan relation and oc-curs in two main cases just in an opposite manner to isMoreSpecificThan relation.

isDifferentFrom relation. This relation occurs when the security concepts specified by the requestor and the provider are semantically disparate. There are two main cases for this relation:

Case 1. It is defined at the level of *AlgorithmSuite* and *SecurityToken* properties of *SecurityBinding* assertions. For example a security token specified by the provider as a username token is considered different from a security token

specified by the requestor as an X509 token. This relation is also defined at the level of all properties of *ProtectionScope* assertion. For example an encrypted part specified by the provider as a Body part is considered different from an encrypted part specified by the requestor as a Header part.
Case 2. *isDifferentFrom* relation is defined at the level of *ProtectionScope* and *SecurityBinding* assertions. A provider security assertion is different from a requestor security assertion if: 1) they point to different security concepts, or 2) they point to the same security concept, and they have at least one *isDifferentFrom* relation at the level of their properties, and their remaining properties are linked with semantic relations of type *isIdenticalTo* or *isEquivalentTo*.

4 Semantic Matching of Security Assertions

In this section, we propose an algorithm for matching provider and requestor security assertions. The matching process consists in checking to what extent each security assertion *RAss* specified in the requestor SP is satisfied by a security assertion *PAss* specified in the provider SP. The matchmaker has to perform two main tasks. Firstly, it must create all possible semantic relations at the level of each pair of provider and requestor assertions. This is done through the execution of the semantic rules presented in the previous section. Secondly, based on the created semantic relations, it must decide the appropriate matching degree for each *PAss-RAss* pair. The final matching degree for a requestor assertion is the highest level of match it has against all of the checked provider assertions. There are four possible matching degrees for a *PAss-RAss* pair: perfect match, close match, possible match, and no match in decreasing order of matching.

Perfect match. A perfect match occurs when *PAss* and *RAss* are connected through *isIdenticalTo* or *isEquivalentTo* relations.

Close match. A close match occurs when *PAss* and *RAss* are connected through *isMoreSpecificThan* relation. For example, suppose that *PAss* and *RAss* point to the *TransportBinding* concept, and *PAss* has a *SecurityHeaderLayout* property having the value "Lax", while *RAss* hasn't a *SecurityHeaderLayout* property, and the other properties of both assertions are identical.

The transport binding specified by the provider is described in more detail than the transport binding specified by the requestor. We assume that the requestor omits specifying the *SecurityHeaderLayout* property because he doesn't care which specific value is used for this property. Therefore, close match is an appropriate matching degree for this example.

Possible match. A possible match is decided in three main cases:

Case 1. *PAss* and *RAss* are connected through *isMoreGeneralThan* relation. This means that the information available can not ensure that *PAss* can perfectly match *RAss*. We assume that a potential partnership between the requestor and the provider can take place if the requestor can obtain additional information or negotiate with the provider.

Case 2. *PAss* is connected to *RAss* through *isLargerThan*, *isStrongerThan*, or *hasTechDiffWith* relations. This means that the incompatibility between the two assertions doesn't negatively affect the security services and levels required in RAss. We assume that a potential partnership between the requestor and the provider can take place if the requestor can strengthen his policy assertion or change some technical properties of his assertion.
Case 3. *PAss* and *RAss* point to the same security concept and have at least one *isMoreGeneralThan*, *isLargerThan*, *isStrongerThan*, or *hasTechDiffWith* relation at the level of their properties, and their remaining properties are linked with semantic relations of type *isIdenticalTo*, *isEquivalentTo*, or *isMoreSpecificThan*. For example, suppose that *PAss* and *RAss* point to the SymmetricBinding concept, but *PAss* has a protection token that is more general than the protection token specified in *RAss*. In addition, *PAss* has an algorithm suite that is identical to the algorithm suite specified in *RAss*. And finally, *PAss* has a Timestamp property that is stronger than the Timestamp property of *RAss*. The two assertions have two heterogeneities that don't rule out the possibility of a match, so it is a possible match case.

No match. No match is decided in two main cases:

Case 1. *PAss* and *RAss* are connected through *isDifferentFrom*, *isSmallerThan*, or *isWeakerThan* relations. For example, suppose that *PAss* points to the *SymmetricBinding* concept, while *RAss* points to the *ProtectionScope* concept. These two assertions specify two semantically unrelated concepts. Therefore, they must be linked by *isDifferentFrom* relation and then a no match must be decided for them.
Case 2. *PAss* and *RAss* point to the same security concept, and they have at least one *isDifferentFrom*, *isSmallerThan*, or *isWeakerThan* relation at the level of their properties.

5 Implementation and Application

In this section, we present implementation details and show how we use our approach to handle the flight reservation use case that was discussed in the second section. We implemented the WS-SP-based ontology WS-SP.owl in OWL-DL [6] semantic language with the Protégé OWL tool [7]. The OWL file can be found at `http://www.redcad.org/members/monia.benbrahim/WS-SP.owl`. Then we created *PSP.owl* and *RSP.owl* ontologies as specializations of WS-SP.owl ontology. *PSP.owl* and *RSP.owl* ontologies must be imported respectively by the provider and the requestor in order to specify their SPs. In addition, we created another ontology called *AssertionMatching.owl* that imports two OWL files representing the provider SP and the requestor SP. This ontology contains the concepts of *PSP.owl* with the prefix `provider` and the concepts of *RSP.owl* with the prefix requestor. In *AssertionMatching.owl* ontology, we added all the new semantic relations previously detailed in section 3.2 as well as all the necessary SWRL rules for their dynamic instantiation. For example the following is the SWRL rule corresponding to the first semantic rule presented in section 3.2:

```
requestor: SignedElement(?RSignedElt) ∧
XPathValue(?RSignedElt, "/S:Enveloppe/S:Body") ∧
provider: SignedPart(?PSignedPart) ∧
PartType(?PSignedPart, "Body")→
isEquivalentTo(?PSignedPart, ?RSignedElt)
```

We also implemented the matching algorithm described in the previous section as a set of SWRL rules and added them to *AssertionMatching.owl* ontology.

The assertions contained in a concrete (requestor/provider) SP are mapped to in-stances of *AssertionMatching.owl* ontology. We use the Jess rule engine [8] to exe-cute the SWRL rules and automatically generate, in a first step, the involved semantic relations between each pair of provider assertion-requestor assertion; and generate, in a second step, the appropriate matching degrees between the assertions.

In order to apply our approach to the use case presented in section 2, we used the *RSP.owl* ontology to transform the requestor SP that was written in WS-SP language into the semantic SP *RSPEx1.owl*[2] . Similarly, based on the *PSP.owl* ontology, the provider SP is semantically described in *PSPEx1.owl*[3]. Then an *AssertionMatching.owl* file that imports *RSPEx1.owl* and *PSPEx1.owl* is generated. After running the Jess engine, a set of semantic relations and matching degrees are created between the provider and the requestor assertions. This is an extract from the new knowledge that was added in the *AssertionMatching.owl* file after the execution of the SWRL rules:

```
<rdf:Description rdf:about="http://.../PSPEx1.owl#PSignedPart">
   <isEquivalentTo rdf:resource="
    "http://.../RSPEx1.owl#RSignedElt"/>
</rdf:Description>
<rdf:Description rdf:about="http://.../PSPEx1.owl#PSymBinding">
    <isStrongerThan rdf:resource=
    "http://.../RSPEx1.owl#RSymBinding"/>
</rdf:Description>
<rdf:Description rdf:about="http://.../PSPEx1.owl#PAss1">
    <PerfectMatch rdf:resource=
    "http://.../RSPEx1.owl#RAss1"/>
</rdf:Description>
```

The application of our semantic approach to match the security assertions specified in the flight reservation use case allows to obtain the expected matching degree at the level of each requestor assertion-provider assertion pair as well as the expected final matching degree at the level of each requestor assertion.

[2] Available at
http://www.redcad.org/members/monia.benbrahim/useCase/RSPEx1.owl
[3] Available at
http://www.redcad.org/members/monia.benbrahim/useCase/PSPEx1.owl

6 Related Work

Several works dealt with the enrichment of WS-Policy with semantics using OWL ontologies. The authors of [18] propose to create the policy assertions based on terms from ontologies instead of XML schema based vocabularies. In [17], Speiser proposes a technique for annotating policy assertions with semantic references as it is done for SAWSDL [9]. Chaari et al. [11] redesign WS-Policy with an ontological representation of concepts and relations. They also define a general QoS ontology and integrate it with the WS-policy ontology. All these contributions just give an idea on how to augment WS-Policy with semantics, but they did not indicate how to define specific domain ontologies such as security ontology. The work described in [15] consists in mapping WS-Policy and related specifications such as WS-SP to OWL-DL. In this work, the authors systematically map each policy assertion to an ontology class. In our approach, we rather take into account the semantic meaning of security policy assertions and make several semantic classifications of security assertions. In addition, we transform some assertions into data type and object properties and not into classes in order to explicit the semantic relationships between assertions.

In the area of Web service security, works such as [12], [14], and [13] propose a security ontology to describe the security requirements and capabilities of Web service providers and requestors. Although these ontologies include concepts related to SOAP message security, they consider neither the message security model defined by WS-Security standard nor the security assertions defined in WS-SP standard. Besides these approaches have the drawback of not being compatible even to WS-Policy. The work proposed by Garcia et al [19] mainly deals with integrity and confidentiality of SOAP messages. A security ontology that considers the WS-Security message security model is proposed and constitutes a foundation to support WS-Policy with semantics. However, this ontology doesn't include concepts to describe many security as-pects such as transport level security and supporting tokens. Moreover, the security concepts used in the ontology are not equivalent to WS-SP assertions.

In addition to being compatible to WS-SP, our approach is distinguished from all pre-vious approaches by the fact that we extend the WS-SP-based ontology with additional semantic relations and semantic rules that support more correct and more pre-cise semantic matching of Web service security policies.

7 Conclusion and Future Work

In this paper, we presented an approach to provide a semantic extension to security assertions of Web services. The approach is based on the transformation of WS-SP into an OWL-DL ontology. We showed, through a top down methodology, how we transform WS-SP assertions into classes and properties in the ontology. Besides, in order to support semantic matching of these assertions,

we extended the domain knowledge with semantic relations that can exist between requestor and provider security concepts. These relations are dynamically instantiated through the execution of SWRL rules. Moreover, we proposed an algorithm for matching security assertions. Our semantic approach supports more correct and more flexible security assertion matching compared to syntactic matching as well as to previous works that combined ontologies and Web service security properties. The on going work is the completion of our matching algorithm in order to decide about the final matching degree between complex security policies containing several security alternatives and assertions. Besides, we plan to develop a tool that automatically transforms a WS-SP policy into our ontological representation.

References

1. OASIS: WS-Security 1.1, http://www.oasis-open.org/specs/
2. OASIS: WS-SecurityPolicy 1.3, http://www.oasis-open.org/specs/
3. WS-Policy 1.5, http://www.w3.org/TR/ws-policy/
4. W3C: SWRL: A Semantic Web Rule Language Combining OWL and RuleML, http://www.w3.org/Submission/SWRL/
5. OASIS: Web Services Security X.509 Certificate Token Profile, http://www.oasis-open.org/specs/
6. W3C: OWL Web Ontology Language Guide, http://www.w3.org/TR/owl-guide/
7. OWL protege Web page, http://protege.stanford.edu/overview/protege-owl.html
8. The Jess engine Web page, http://www.jessrules.com/
9. W3C: Semantic Annotations for WSDL and XML Schema, http://www.w3.org/TR/sawsdl/
10. Ben Brahim, M., Ben Jemaa, M., Jmaiel, M.: Security Mapping to Enhance Matching Fine-Grained Security Policies. In: Zavoral, F., Yaghob, J., Pichappan, P., El-Qawasmeh, E. (eds.) NDT 2010, Part I. CCIS, vol. 87, pp. 183–196. Springer, Heidelberg (2010)
11. Chaari, S., Badr, Y., Biennier, F.: Enhancing web service selection by qos-based ontology and ws-policy. In: Proceedings of the 2008 ACM Symposium on Applied Computing, pp. 2426–2431. ACM (2008)
12. Denker, G., Kagal, L., Finin, T.W., Paolucci, M., Sycara, K.: Security for DAML Web Services: Annotation and Matchmaking. In: Fensel, D., Sycara, K., Mylopoulos, J. (eds.) ISWC 2003. LNCS, vol. 2870, pp. 335–350. Springer, Heidelberg (2003)
13. He, Z., Wu, L., Hong, Z., Lai, H.: Semantic security policy for web service. In: Proceedings of the IEEE International Symposium on Parallel and Distributed Processing with Applications, pp. 258–262. IEEE Computer Society (2009)
14. Kim, A., Luo, J., Kang, M.: Security Ontology for Annotating Resources. In: Meersman, R., Tari, Z. (eds.) OTM 2005, Part II. LNCS, vol. 3761, pp. 1483–1499. Springer, Heidelberg (2005)
15. Kolovski, V., Parsia, B., Katz, Y., Hendler, J.: Representing Web Service Policies in OWL-DL. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 461–475. Springer, Heidelberg (2005)

16. Ono, K., Nakamura, Y., Satoh, F., Tateishi, T.: Verifying the consistency of security policies by abstracting into security types. In: Proceedings of the 2007 IEEE International Conference on Web Services, pp. 497–504. IEEE Computer Society (2007)
17. Speiser, S.: Semantic annotations for ws-policy. In: Proceedings of the 2010 IEEE International Conference on Web Services, pp. 449–456. IEEE Computer Society (2010)
18. Verma, K., Akkiraju, R., Goodwin, R.: Semantic matching of web service policies. In: Proceedings of the Second Workshop on Semantic and Dynamic Web Processes, pp. 79–90 (2005)
19. Zuquim Guimaraes Garcia, D., Beatriz Felgar de Toledo, M.: Ontology-based security policies for supporting the management of web service business processes. In: Proceedings of the 2th IEEE International Conference on Semantic Computing, pp. 331–338. IEEE Computer Society (2008)

Quality Prediction in Service Composition Frameworks

Benjamin Klatt, Franz Brosch, Zoya Durdik, and Christoph Rathfelder

FZI Karlsruhe, Haid-und-Neu-Str. 10-14, 76131 Karlsruhe, Germany
{klatt,brosch,durdik,rathfelder}@fzi.de

Abstract. With the introduction of services, software systems have become more flexible as new services can easily be composed from existing ones. Service composition frameworks offer corresponding functionality and hide the complexity of the underlying technologies from their users. However, possibilities for anticipating quality properties of composed services before their actual operation are limited so far. While existing approaches for model-based software quality prediction can be used by service composers for determining realizable Quality of Service (QoS) levels, integration of such techniques into composition frameworks is still missing. As a result, high effort and expert knowledge is required to build the system models required for prediction. In this paper, we present a novel service composition process that includes QoS prediction for composed services as an integral part. Furthermore, we describe how composition frameworks can be extended to support this process. With our approach, systematic consideration of service quality during the composition process is naturally achieved, without the need for detailed knowledge about the underlying prediction models. To evaluate our work and validate its applicability in different domains, we have integrated QoS prediction support according to our process in two composition frameworks – a large-scale SLA management framework and a service mashup platform.

1 Introduction

In recent years, service-oriented systems have gained more and more attention from software providers and consumers. Reduced ramp-up-time, infrastructure cost and management advantages reside on both sides [1]. Meanwhile, service providers do not only offer single services but also composed services on demand, according to individual customer requests. *Service composition frameworks* are emerging [2,3] allowing for intuitive creation of composed services without requiring expert knowledge about the underlying technologies.

Apart from functional requirements, *Quality of Service* (QoS) properties – such as performance and reliability – are increasingly important to assure user acceptance. Anticipating QoS properties of dynamically composed services is challenging, as it is typically not feasible to create and test each possible composition that a customer might request in a laboratory or field environment. A potential solution to this problem is the use of *model-based software quality*

G. Pallis et al. (Eds.): ICSOC 2011, LNCS 7221, pp. 131–146, 2012.

prediction [4], which does not require the actual execution of services in order to assess them. Rather, it uses simulations or analytical calculations to predict service quality based on the specification of the service architecture and its quality-relevant factors.

However, the missing integration of predictive capabilities in the composition frameworks is a major obstacle to the use of quality prediction approaches. If prediction is applied in isolation, the manual specification of the required input information is laborious and sometimes even impossible due to missing knowledge about the internals of the service architecture. Furthermore, many prediction approaches require to understand the specifics of the employed prediction models. Especially if a customer is interested choosing one of several possible alternatives for service composition, evaluating all of them through repeated manual prediction is highly impracticable.

To overcome this problem, we present a *quality-aware service composition process* in this paper. This process includes QoS prediction for composed services as an integral part and to support the decision between multiple composition alternatives. Furthermore, we describe how composition frameworks can be extended to support our process. The description includes a set of components and interactions that realize the envisioned functionality. We validate our work in two case studies where we integrate QoS prediction support in a large-scale SLA management framework and a service mashup platform. In these case studies, QoS prediction is based on the *Palladio Component Model* (PCM, see [5]).

The contributions of this paper are (i) a quality-aware service composition process, (ii) a schema for integrating corresponding QoS prediction support in service composition frameworks, and (iii) two case studies using a reference implementation based on the PCM.

The remainder of this paper is structured as follows: First, Section 2 provides an introduction to model-based software quality prediction and service composition as relevant foundations for our approach. Our recommended process is introduced in Section 3, followed by the integration in composition frameworks in Section 4. Section 5 describes the provided reference implementation and the two case studies before conclusions are drawn in Section 6.

2 Foundations and Related Work

The work presented in this paper is based on two major research areas. On the one hand, model-based software quality prediction provides a foundation for the predictive capabilities to be integrated in the process. On the other hand, service compositions provide new flexibility for software creation, but also challenges regarding the consideration of QoS properties. In the following, both areas are presented in more detail, followed by a discussion of related work to our approach.

2.1 Services and Service Compositions

Software services are self-describing entities. They encapsulate application functionality and information resources, and make them available through

programmatic interfaces [6]. This encapsulation empowers service compositions, which orchestrate a set of existing services to a new service or application, to provide more complex functionality [7]. The composition of services is described as a workflow using a modelling or programming language, such as the Business Process Execution Language (BPEL) [8] or any kind of glue code. Recent developments aim to ease the service composition to enable people with little or none programming or engineering skills to do this. Service composition frameworks and platforms provide non-programmers with flexible and intuitive general-purpose development environments [9], such as IBM Mashup Center [10], Yahoo! Pipes [11], and DreamFace [12].

The value of composed services is not only assessed in terms of functionality, but also according to their QoS properties, such as response time, throughput or reliability [13,7]. Service Level Agreements (SLAs) have been introduced to contractually define the terms and conditions of a service delivered to a customer. SLA terms may relate to functional and QoS properties, as well as qualifying conditions (such as expected workload). The major constituent of an SLA is its QoS-related information [6].

2.2 Model-Based Software Quality Prediction

Model-based software quality prediction may relate to different QoS properties, such as performance and reliability. Regarding performance, Smith et al. established the term *Software Performance Engineering* (SPE) combining performance analyses with a focus on software design [14]. Meanwhile, the performance engineering community has developed numerous approaches with improved accuracy and usability as surveyed in [15]. Similarly, various approaches exist to predict reliability of software systems and services [16].

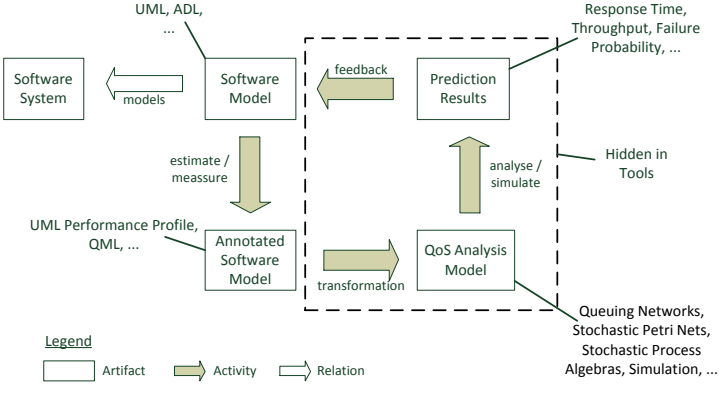


Fig. 1. OMG Quality Prediction Process [17]

A general process for model-based quality prediction has been defined by the Object Management Group (OMG) and is presented in Figure 1 [17]. The starting point of the process is a model of the software itself. In the next step,

quality-related annotations are added using either a specific UML profile, such as the UML profile for Schedulability, Performance and Time (UML-SPT) [18], the UML profile for QoS [19], the UML-MARTE profile [20], or a quality-specific modelling language, such as KLAPER [21] or the Palladio Component Model [5]. The annotated model is then transformed to a QoS analysis model, such as a queueing network [22], Petri net [23], Markov chain [24] or Bayesian network [25], for the prediction itself. The prediction results may be further processed or aggregated if necessary, and returned as a feedback to the modeller, allowing to assess expected QoS levels, and to decide between architectural alternatives.

2.3 Related Work

Various aspects of QoS-aware service composition have been addressed in literature [7], including service discovery and optimization strategies. Klein et al. [26] used linear programming techniques to provide a set of probabilistic service selection policies to chose services at runtime. However, their approach requires runtime information, they are focused on a long-term optimization, and they assume that services can be chosen automatically. Canfora et al. [13] use genetic algorithms to optimize the service selection within a composition. Their algorithm considers QoS as well as cost parameters. However, they are focused on runtime service selection and optimization and do not consider SLA-based quality prediction of service compositions in advance. Garcia et al. [27] describe a framework for service selection based on functional and non-functional properties. However, they are focused on single service discovery and do not consider QoS prediction for service compositions.

A closely related work to our approach is a model-based analysis of service compositions utilizing UML activity models by Gallotti et al. [28]. It describes an assessment of service execution time and reliability properties that can be applied at design time. The activity models which describe the composition are enriched by exploiting a subset of the MARTE UML profile [20]. However, the approach does not consider the automated integration in a service composition framework transparently to the service composer. It also does not process available SLAs to extract quality-related information to merge them into a quality prediction model.

In a preliminary work [29], we describe the details of the QoS prediction integration in the service mashup platform that constitutes our second case study in this paper (see Section 5.3). However, we do not derive a generalized prediction concept and composition process as we do in this paper.

3 Quality-Aware Service Composition

This section introduces our approach of a service composition process with integrated quality prediction. The process describes general steps to create composed services that fulfil given quality requirements. It is designed to consider the generic quality-influencing factors of service compositions, which are presented in Figure 2. The first factor is the service composition workflow (i.e. control and data flow and interactions between involved services) that can have a big impact

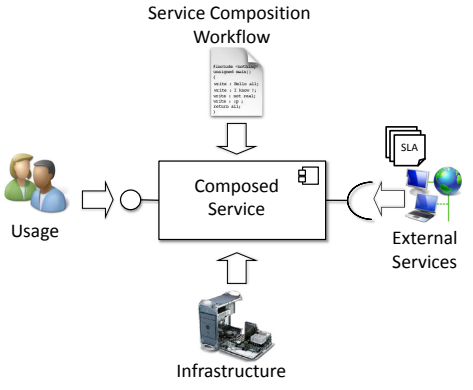


Fig. 2. Service Quality Impact Factors

on the quality of the composed service. The second factor is second, QoS properties of externally required services (e.g., response time and throughput) that can have a significant impact, too. The third factor is the infrastructure used to execute the composed service. Characteristics such as the available memory, storage throughput, and cpu performance are relevant for the service execution. Finally, the service usage influences the service quality – for example, a service might perform well for a small user group with a low arrival rate, but it might show a poor performance for a large user group with frequent service requests. Usage characteristics of interest include the expected workload and invocation behaviour, including call frequencies and input data. External service quality, infrastructure quality and service usage profiles can be contractually specified through SLAs and, thus, be automatically taken into account for QoS prediction. Notice that, in order to achieve accurate prediction results, the underlying QoS prediction method should be chosen in a way such that it explicitly takes all described factors into account.

Figure 3 presents an UML activity diagram describing our proposed generic process. This process is conducted by a user of a service composition framework, namely a service composer. The right hand side of the figure shows artefacts related to external services used by the composed service. The left hand side shows QoS requirements that the composed service has to provide. In the central area, the main steps of the process are shown.

The process starts with the specification of the service composition workflow, which can either be created on demand or chosen from a set of predefined workflows. In both cases, the service composition workflow includes dependencies to external software or infrastructure services required to realize and execute the composed service. Based on the specified workflow, the service composer selects external service instances that match the referenced descriptions. There might be multiple instances available per description with different QoS properties, as specified in the corresponding SLAs.

The service selection and the usage profile specification provide SLAs which contain all relevant information to automatically derive a quality prediction

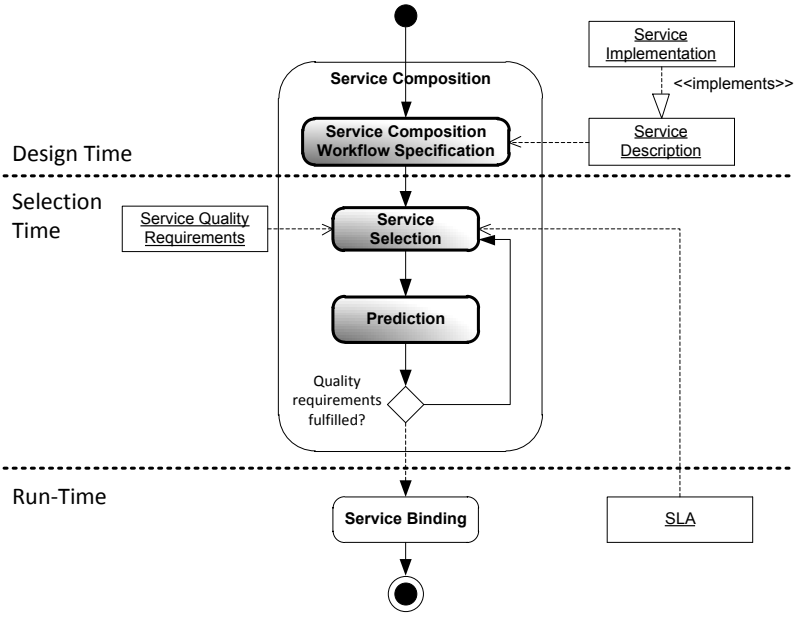


Fig. 3. Service Composition Process: Steps and Artefacts

model. Based on this prediction model, an automated quality prediction can be performed to provide feedback about the ability of the composition to satisfy the customer's quality requirements. Due to the high envisioned degree of automation, the service composer does not need any expert knowledge about the underlying prediction models.

The prediction results are compared with the initial quality requirements. If they are not satisfied, the service composer can select other external service instances with different QoS properties and repeat the prediction. If applicable, the composer can also change the set of considered quality requirements or the workflow specification itself to adapt the preconditions of the scenario. Service selection and prediction may be repeated multiple times, without actually instantiating and executing the composed service, until a satisfying configuration is found. Eventually, the process moves on to the instantiation of the composed service, indicated in Figure 3 through the binding step.

4 Framework Integration

This section discusses how service composition frameworks can be enhanced with predictive capabilities according to our quality-aware service composition process. The prediction functionality is provided by a *Service Quality Prediction* (SQP) component which can be integrated in a composition framework. The details of concrete SQP implementations may vary between different frameworks; however, the conceptional specification as provided here is still valid.

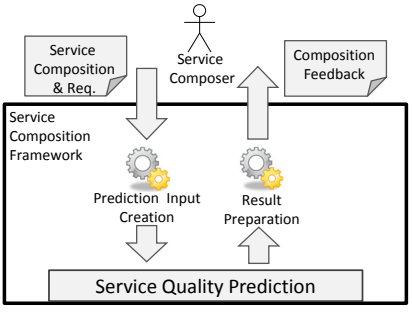


Fig. 4. SQP Integration in Service Composition Frameworks

Figure 4 shows how the SQP component is integrated into a composition framework. An important aspect of the integration is the transparency of the QoS prediction details to the user of the framework, namely the service composer. From the composer's point of view, the inputs for the prediction are the specification of a composed service (i.e. the composition workflow specification and the selected external service instances), the envisioned service usage and a set of quality requirements. These inputs are specified in a user-oriented way, through the interface between the framework and the composer. Similarly, prediction results are returned as a feedback to the composer in a form that is most understandable and meaningful to him (e.g., a graphical presentation of predicted QoS metrics or a simple statement if the given QoS requirements are met).

Internally, the composer's inputs are translated to a valid input format for prediction. This translation may include several aspects such as automated interpretation of machine-readable SLA contents and automated creation or adaptation of prediction models. Similarly, prediction results may have to undergo further processing such as aggregation or translation into user-understandable result metrics before they are presented to the service composer.

This decoupling between user- or framework-specific SLA and service description format from one side, and the internal prediction specific representation from another side, supports the usage of proprietary and standard formats (e.g., wsdl, ws agreement, and BPMN). For example, the SLA@SOI framework described in Section 5.2 encapsulates multiple standards. However, even as intermediary format no specific standard could be used as they all have different focus. For example, the ws agreement standard focuses more on the negotiation than the description of quality properties.

Figure 5 presents further details of the internal structure and prediction workflow of the SQP component. The prediction input is delivered to SQP in the specific format suited for the applied prediction method by an external *Prediction Input Creation* component. This component acts as an adapter between SQP and the surrounding framework. Typically, the input includes the prediction model itself and further configuration settings, such as required simulation depth or maximum simulation time.

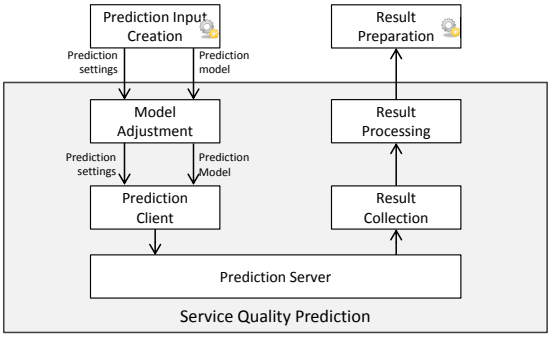


Fig. 5. Internal SQP Structure and Workflow

If the available input information is not sufficient to fully create the prediction model, an alternative solution is that SQP loads a predefined base model and adjusts certain parameters of this model according to particular prediction requests. This may be done by an SQP-internal *Model Adjustment* subcomponent. Furthermore, the prediction itself, done by simulation or by analytical solving, may be a time- and resource-consuming task, which should be outsourced to dedicated prediction servers. To this end, SQP may include a *Prediction Client* that sends its requests to a physically remote server. In this case, SQP is a distributed subsystem on its own rather than a single component. Further SQP-internal steps include the collection of prediction results returned by the server and potentially their aggregation (e.g. deriving mean values or percentiles from prediction probability distributions). Finally, the results are returned to the surrounding framework which may further process them for presentation to the service composer.

5 Case Studies

To demonstrate the applicability of our approach, we have integrated QoS prediction according to our described process (see Section 3) in two service composition frameworks. In the following, we first provide an introduction to the prediction method we used for realizing the QoS prediction (Section 5.1). Then, we discuss the two service composition frameworks and how we extended them (Sections 5.2 and 5.3).

5.1 QoS Prediction with the Palladio Component Model

As a prediction method for our case studies, we have chosen the Palladio Component Model (PCM) [5]. While traditionally being tailored towards component-based software architectures, the PCM can also be used to model and predict the performance and reliability of composed services, explicitly taking into account all quality-influencing factors described in Section 3. Its enhanced application of model driven development (MDD) techniques provides a basis for automated

transformation of the service composition inputs into prediction models such as queueing networks (QN) for performance simulation or Markov chains for reliability analysis. Realized as stand-alone tooling, the PCM modelling and prediction capabilities have been successfully evaluated in a number of research and industrial case studies [30,31,32,33].

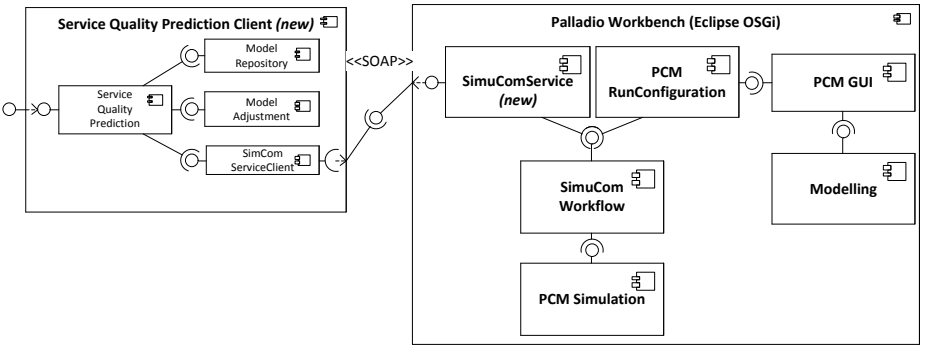


Fig. 6. PCM Extension for QoS Prediction of Service Compositions

To achieve a transparent integration of PCM-based QoS prediction into service composition frameworks, we have extended the existing *PCM Workbench* to offer automated prediction "as a service". We focus on the performance prediction through QN-based simulation. Figure 6 gives an overview of the PCM components relevant in our context, including existing and newly added ones. For stand-alone QoS prediction, software architects use the *PCM GUI* component to create a design-oriented model of the component-based architecture. The *SimuComWorkflow* transforms the architectural model to a QN and controls the actual prediction carried out through the *PCM Simulation*. The *PCMRunConfiguration* offers various configuration parameters for the simulation. A new *SimuComSerivce* (a SOAP-based web service) has been developed to enable programmatically triggered predictions as an alternative to the manual operation. The service can be remotely invoked and returns aggregated prediction results to its callers.

While the extended PCM Workbench as described here provides the basis for applying the PCM in both case studies, the client side required two individual implementations tailored to the needs of each case study. Figure 6 shows the implementation provided for the first case study (see Section 5.2) as an example. The implementation includes a set of components offering a single SQP interface to the surrounding service composition framework. An internal *Service Quality Prediction* component controls the SQP workflow. A *ModelRepository* stores and loads PCM-based quality prediction models. The *ModelAdjustment* adapts models according to individual prediction requests. The *SimuComServiceClient* encapsulates the remote communication with the *SimuComService* and includes functionality for results collection and processing as shown in Figure 5.

5.2 Case Study I: SLA Management Framework

In the first case study, we integrated QoS prediction in a large scale SLA management framework developed within the SLA@SOI research project [2]. The central goal of the project is to enable a transparent SLA management across the whole business and IT stack. It aims to automate SLA negotiation and to provision, deliver and monitor services in a unified manner with predictable quality attributes that can be enforced at run-time. Based on our approach, service performance and reliability prediction can be applied during the SLA negotiation process to support service providers in creating feasible SLA offers, as illustrated in Figure 7. In addition to the work described in [34], this integration has been further enhanced and was extended with the experience gathered from the second case study described below.

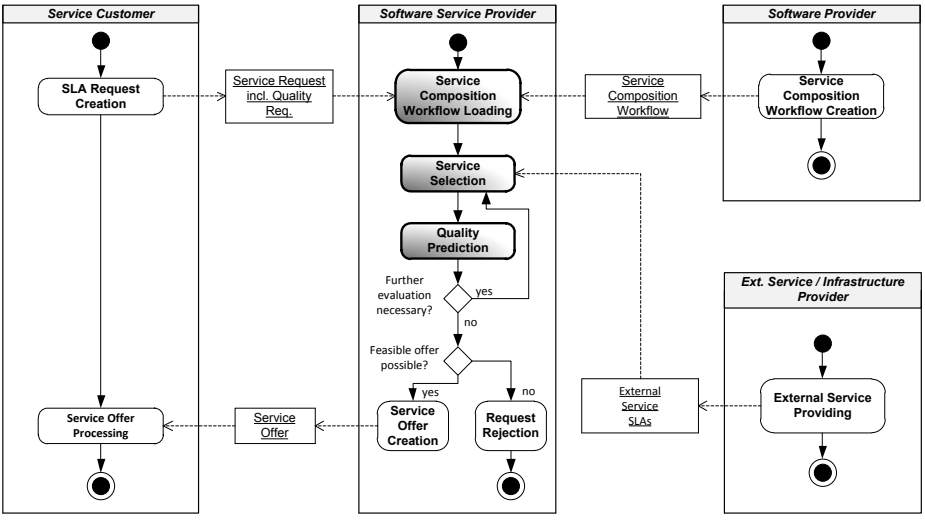


Fig. 7. Quality-Aware Service Composition for SLA@SOI

In the SLA@SOI scenario, a *Software Service Provider* offers a set of software services to *Service Customers* with non-functional property ranges. During SLA negotiation, a service provider and a customer establish a concrete SLA with fixed values for the available property ranges. The provider in turn may act as a customer of other service providers, making SLA negotiation a multi-level process.

The negotiation is initiated by the service customer, who issues an SLA request to the provider. Such a request contains a selection of service operations, as well as a required service quality for a maximum frequency of service invocations. It is the responsibility of the provider to determine if the service can be offered under the requested conditions and for a reasonable price. To this end, he starts our identified general quality prediction process as highlighted

in Figure 7. In the SLA@SOI scenario, he first loads the already existing composition workflow provided by the *Software Provider* for the requested service. Then, he selects a certain set of SLAs of the required external services. The latter step includes the selection of appropriate services from external providers, as well as the negotiation of concrete service quality, usage and pricing for these external services. As a third step, the service provider uses the service quality prediction to determine the expected performance and reliability of the target services. Finally, he evaluates the prediction results. If the results satisfy the quality requirements of the SLA request, he returns a concrete SLA offer back to the requesting customer who may accept or decline this offer. If prediction shows that the quality requirements cannot be met, the service provider can still return a counter-offer containing the predicted quality to the customer, and the customer can still accept this offer. Alternatively, the provider can evaluate other service configurations. If he cannot create any feasible offer in response to the original request, he rejects the request and no service offer is established.

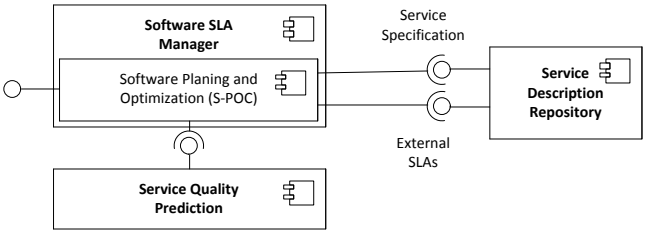


Fig. 8. SLA@SOI Framework Support for Automated SLA Negotiation

Figure 8 shows the parts of the SLA management framework that are involved in the SLA negotiation. QoS prediction is provided by the SQP component as introduced in Section 4. The service description repository comprises the SLA registry and a registry for the available service composition workflows. A *Software Planning and Optimization Component* (S-POC) as part of a *Software SLA Manager* automatically selects service configurations to be predicted on behalf of the software service provider. The SLA negotiation process is not part of the approach presented in this paper; further information about this topic can be found in [35]. The automation removes the burden of manually finding good configurations from the provider, and hides the complexity of the underlying multi-level SLA negotiation process.

In the context of the SLA@SOI project, the capabilities of the SLA management framework have been demonstrated and validated on several industrial use cases covering different application domains, such as ERP hosting, public telecommunication services, Enterprise IT management and eGovernment. Service performance and reliability prediction has been specifically conducted in the ERP hosting and eGovernment use cases, as well as an open source demonstrator centred around a sales services solution. Further details about these use cases have been published in the SLA@SOI deliverables at [2]. The results show

that our approach could be applied to successfully integrate quality predictions in the SLA@SOI framework.

5.3 Case Study II: Service Mashup Platform

In the second case study, we integrated QoS prediction support in the service mashup platform envisioned by and realized within the COCKTAIL research project [3]. The platform provides methods and tools for the creation, composition and commercial marketing of light-weight services, which are mostly referred to as *service mashups*. Integrated aspects of the platform include service enabling and operation, logging, user management and accounting based on flexible pricing models. Our extension of the platform with QoS prediction capabilities enhances service composition by informing service composers about the quality properties to be expected for their specified compositions.

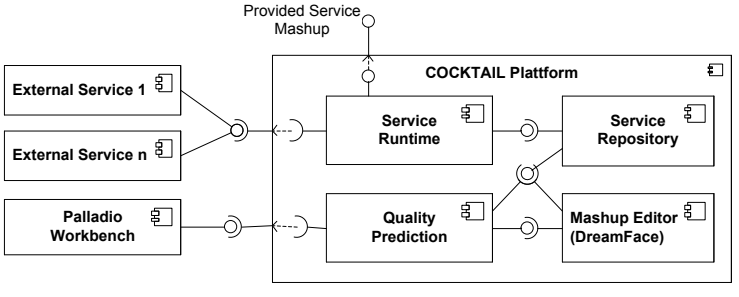


Fig. 9. COCKTAIL Service Platform (excerpt)

Figure 9 shows the parts of the platform involved in the service composition, namely *ServiceRepository*, *MashupEditor*, *ServiceRuntime* and *QualityPrediction*. The *ServiceRepository* contains information about the services that are available on the platform and their properties, such as the type of service, input and output parameters, pricing models and quality attributes. The *MashupEditor* enables the composition of services available in the platform. An existing *MashupEditor* named DreamFace [12] has been adopted for the use in the platform. The *ServiceRuntime* component provides the execution environment for service operation. The *QualityPrediction* component provides the QoS prediction support in the platform. It is invoked by the *MashupEditor* and enables the prediction of the service quality according to the given service composition workflow specification, the specified infrastructure and the service usage profile.

Figure 10 presents the top-level process of service composition and quality prediction in COCKTAIL. There are four roles involved in the service composition: *Service Provider*, *Service Enabler*, *Service Composer* and *Service Customer*. Service Providers develop services for the platform and provide service descriptions including quality-relevant characteristics. Service Enablers store the services in the *ServiceRepository* and enable their operation. Potential customers issue service requests which may be associated with specific functional and QoS

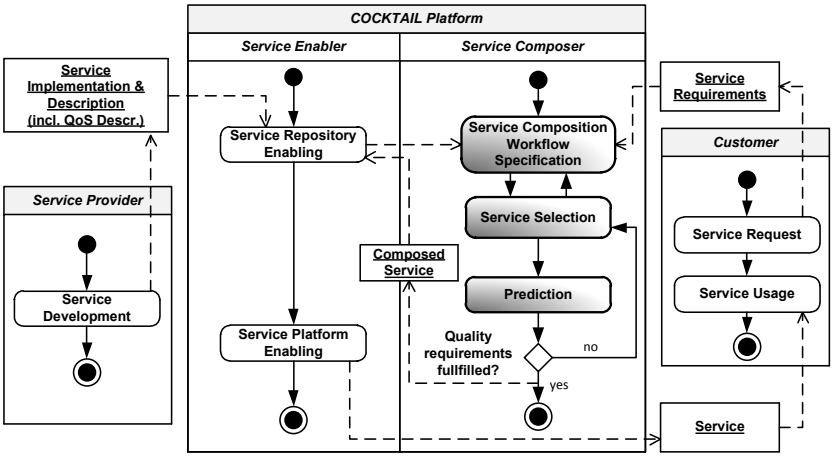


Fig. 10. Quality-Aware Service Composition for COCKTAIL

requirements. If a Service Customer's request cannot be satisfied with an already existing service, the request is sent to a Service Composer to create a corresponding new service. As highlighted in Figure 10, this is done according to the general process proposed in this paper. First, the Service Composer specifies a composition workflow and selects services from the repository. Specific to COCKTAIL, those two steps are done iteratively within the *MashupEditor*. Next, the prediction is conducted to get feedback about the quality properties of the service. This feedback is used to assess the service based on its quality and cost properties. As soon as a satisfying service configuration is found, the composed service is created and stored in the *ServiceRepository* bundled with its specification and quality description. The Service Enabler publishes the service to be operated in the platform and used by the customer.

The COCKTAIL service mashup platform with integrated quality prediction according to our approach has been successfully demonstrated and evaluated on a CRM (Customer Relationship Management) scenario [29].

6 Conclusions and Outlook

In this paper, we have introduced a quality-aware service composition process and its integration in service composition frameworks. Our process is designed to consider the quality-influencing factors of composed services, namely the service composition workflow, the external software and infrastructure service quality and the usage profile. The proposed framework integration intuitively allows for consideration of QoS properties during service composition, without the need for expert knowledge about the underlying prediction models. We demonstrated the applicability of our approach in two case studies, where we integrated QoS prediction support according to our process in two service composition frameworks.

The implementation includes performance and reliability prediction based on the Palladio Component Model (PCM).

As shown in the case studies, there is a need for an integrated quality-aware service composition done by service composers who are no experts in quality prediction. Our approach enables the integration of existing QoS prediction approaches in service composition frameworks to support this requirement. Furthermore, we structured the service composition process in a way that ensures all required information to be available when needed, such as SLAs of external services required for the prediction.

Currently, we are integrating our implemented PCM extensions into the existing PCM tool set, which is available as an open source project at [36]. Furthermore, we are planning to support and evaluate additional prediction techniques, and we are working on an enhanced integration of quality predictions in composition frameworks. As part of this we will investigate possibilities of using run-time information as an additional information source for the predictions. These improvements shall further enhance and establish a quality-aware service engineering with predictable and readily negotiable QoS levels.

Acknowledgments. This work was supported by the European Commission (grant No. FP7-216556) and by the German Federal Ministry of Education and Research (grant No. 01BS0822). The authors are thankful for this support.

References

1. Kingstone, S.: Understanding total cost of ownership of a hosted vs. premises-based crm solution. Yankee Group Report June 2004 (2004)
2. SLA@SOI: Project website (June 2011), http://sla-at-soi.eu/
3. COCKTAIL: Project website (June 2011), http://www.cocktail-projekt.de/
4. Balsamo, S., Marco, A.D., Inverardi, P., Simeoni, M.: Model-based performance prediction in software development: A survey. IEEE Transactions on Software Engineering 30, 295–310 (2004)
5. Reussner, R., Becker, S., Burger, E., Happe, J., Hauck, M., Koziolek, A., Koziolek, H., Krogmann, K., Kuperberg, M.: The Palladio Component Model. Technical report, Karlsruhe (2011)
6. Zeng, L., Benatallah, B., Ngu, A.H.H., Dumas, M., Kalagnanam, J., Chang, H.: Qos-aware middleware for web services composition. IEEE Trans. Softw. Eng. 30, 311–327 (2004)
7. Strunk, A.: Qos-aware service composition: A survey. In: European Conference on Web Services (2010)
8. Margolis, B., Sharpe, J.: SOA for the business developer: concepts, BPEL, and SCA. MC Press (2007)
9. Soi, S., Baez, M.: Domain-Specific Mashups: From All to All You Need. In: Daniel, F., Facca, F.M. (eds.) ICWE 2010. LNCS, vol. 6385, pp. 384–395. Springer, Heidelberg (2010)
10. IBM: Mashup center (June 2011), http://greenhouse.lotus.com/
11. Yahoo!: Yahoo! pipes (June 2011), http://pipes.yahoo.com/pipes
12. DreamFace: Mashup editor (June 2011), http://dreamface-interactive.com

13. Canfora, G., Di Penta, M., Esposito, R., Villani, M.L.: An approach for qos-aware service composition based on genetic algorithms. In: Proceedings of GECCO 2005 (2005)
14. Smith, C.U.: Performance Engineering of Software Systems. Addison-Wesley Longman Publishing Co., Inc., Boston (1990)
15. Koziolek, H.: Performance Evaluation of Component-based Software Systems: A Survey. Performance Evaluation 67(8), 634–658 (2010)
16. Goseva-Popstojanova, K., Trivedi, K.S.: Architecture-based approach to reliability assessment of software systems. Performance Evaluation 45 (May 2001)
17. Becker, S.: Coupled Model Transformations for QoS Enabled Component-Based Software Design. PhD thesis, University of Oldenburg, Germany (March 2008)
18. Object Management Group (OMG): UML Profile for Schedulability, Performance and Time (January 2005)
19. Object Management Group (OMG): UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms (May 2005)
20. Object Management Group (OMG): UML Profile for Modeling and Analysis of Real-Time and Embedded systems (MARTE) (May 2006)
21. Grassi, V., Mirandola, R., Sabetta, A.: From Design to Analysis Models: A Kernel Language for Performance and Reliability Analysis of Component-based Systems. In: WOSP 2005. ACM Press (2005)
22. Sharma, V.S., Jalote, P., Trivedi, K.S.: A performance engineering tool for tiered software systems. In: Proceedings of the 30th Annual International Computer Software and Applications Conference, vol. 1. IEEE Computer Society (2006)
23. Kounev, S.: Performance modeling and evaluation of distributed component-based systems using queueing petri nets. IEEE Trans. Softw. Eng. 32, 486–502 (2006)
24. Reussner, R.H., Schmidt, H.W., Poernomo, I.H.: Reliability prediction for component-based software architectures. Journal of Systems and Software 66(3) (2003)
25. Roshandel, R., Medvidovic, N., Golubchik, L.: A Bayesian Model for Predicting Reliability of Software Systems at the Architectural Level. In: Overhage, S., Ren, X.-M., Reussner, R., Stafford, J.A. (eds.) QoSA 2007. LNCS, vol. 4880, pp. 108–126. Springer, Heidelberg (2008)
26. Klein, A., Ishikawa, F., Honiden, S.: Efficient heuristic approach with improved time complexity for qos-aware service composition. In: The 9th International Conference on Web Services, ICWS 2011 (2011)
27. García, J.M., Ruiz, D., Ruiz-Cortés, A., Martín-Díaz, O., Resinas, M.: An Hybrid, QoS-Aware Discovery of Semantic Web Services Using Constraint Programming. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749, pp. 69–80. Springer, Heidelberg (2007)
28. Gallotti, S., Ghezzi, C., Mirandola, R., Tamburrelli, G.: Quality Prediction of Service Compositions through Probabilistic Model Checking. In: Becker, S., Plasil, F., Reussner, R. (eds.) QoSA 2008. LNCS, vol. 5281, pp. 119–134. Springer, Heidelberg (2008)
29. Durdik, Z., Drawehn, J., Herbert, M.: Towards automated service quality prediction for development of enterprise mashups. In: 5th International Workshop on Web APIs and Service Mashups @ ECOWS 2011, Lugano, Switzerland (September 2011) (to appear)
30. Huber, N., Becker, S., Rathfelder, C., Schweflinghaus, J., Reussner, R.: Performance Modeling in Industry: A Case Study on Storage Virtualization. In: International Conference on Software Engineering (ISCE), Software Engineering in Practice Track, pp. 1–10. ACM (2010)

31. Rathfelder, C., Kounev, S., Evans, D.: Capacity Planning for Event-based Systems using Automated Performance Predictions. In: 26th IEEE/ACM International Conference on Automated Software Engineering (2011) (to appear)
32. Brosig, F., Kounev, S., Krogmann, K.: Automated Extraction of Palladio Component Models from Running Enterprise Java Applications. In: Proceedings of the 1st International Workshop on Run-time Models for Self-managing Systems and Applications. ACM (2009)
33. Martens, A., Becker, S., Koziolek, H., Reussner, R.: An Empirical Investigation of the Applicability of a Component-Based Performance Prediction Method. In: Thomas, N., Juiz, C. (eds.) EPEW 2008. LNCS, vol. 5261, pp. 17–31. Springer, Heidelberg (2008)
34. Comuzzi, M., Kotsokalis, C., Rathfelder, C., Theilmann, W., Winkler, U., Zacco, G.: A Framework for Multi-level SLA Management. In: Dan, A., Gittler, F., Toumani, F. (eds.) ICSOC/ServiceWave 2009. LNCS, vol. 6275, pp. 187–196. Springer, Heidelberg (2010)
35. Kotsokalis, C., Yahyapour, R., Gonzalez, M.A.R.: Sami: The sla management instance. In: Proceedings of ICIW 2010 (September 2010)
36. Palladio: Project website (June 2011), http://www.palladio-simulator.com/

ECMAF: An Event-Based Cross-Layer Service Monitoring and Adaptation Framework

Chrysostomos Zeginis, Konstantina Konsolaki,
Kyriakos Kritikos, and Dimitris Plexousakis

Department of Computer Science, University of Crete, Greece
and Information Systems Laboratory, ICS-FORTH, Greece
{zegchris,konsolak,kritikos,dp}@ics.forth.gr

Abstract. Although several techniques have been proposed towards monitoring and adaptation of Service-Based Applications (SBAs), few of them deal with cross-layer issues. This paper proposes a framework, able to monitor and adapt SBAs across all functional layers. This is achieved by using techniques, such as event monitoring and logging, event-pattern detection, and mapping between event patterns and appropriate adaptation strategies. In addition, a taxonomy of adaptation-related events and a meta-model describing the dependencies among the SBA layers are introduced in order to "capture" the cross-layer dimension of the framework. Finally, a specific case study is used to illustrate its functionality.

Keywords: event, monitoring, adaptation, cross-layer, service, pattern, Event-Calculus, non-functional.

1 Introduction

Web services are an emerging technology attracting a lot of attention from both academia and industry in recent years. Thus, more and more businesses adopt them to facilitate and automate their business processes. However, once services and business processes become operational, several emerging issues must be considered throughout the life-cycle of a Service-Based Application (SBA), such as the ones concerning service monitoring and adaptation. These two processes are tightly connected and result in improving and customizing non-performing services, so as to adapt to the context changes and satisfy new requirements.

As far as monitoring is concerned, SBAs need to be managed and monitored, so that stakeholders have a clear view of how services perform within their operational environment, take management decisions, and perform control actions to modify and adjust their behavior. In [5], Web service monitoring is defined as the process of collecting and reporting relevant information about Web service execution and evolution. Many monitoring approaches have been proposed, most of them focusing on measuring QoS metrics, gathered on the basis of SLAs, given that an SLA dictates what are the service obligations in terms of performance. The measured metrics are then compared with the corresponding SLA bounds to detect possible violations.

G. Pallis et al. (Eds.): ICSOC 2011, LNCS 7221, pp. 147–161, 2012.

Furthermore, the dynamic and ever-changing nature of the business and physical environment requires Web services to be highly reactive and adaptive to the changes and variations they are subjected to. They should be equipped with mechanisms to ensure that they can adapt to meet changing requirements. Such changes are identified, detected, and foreseen in the running SBA during the monitoring process. In [10] Web service adaptation is defined as a process of modifying SBAs so as to satisfy new requirements dictated by environmental changes on the basis of adaptation strategies designed by the system integrator.

However, it is critical that monitoring and adaptation occur across all the SBA functional layers, as they are adopted by many projects, such as the S-cube (http://www.s-cube-network.eu) and SLA@SOI (http://sla-at-soi.eu); namely the Business Process Management layer (BPM), the Service Composition and Coordination layer (SCC) and the Service Infrastructure layer (SI). These layers are closely related and many dependencies exist among them. A thorough review of current monitoring and adaptation techniques [25] reveals that these techniques are very fragmented. Although current approaches cover a wide spectrum of monitoring and adaptation [1,9,2,3], few of them deal with cross-layer issues. The latter ones either do not provide a concrete solution to the problem [12], or do not take into consideration all SBA layers [9,22], or do not elaborate on monitoring issues [26].

In this paper, a cross-layer monitoring and adaptation framework is outlined that is based on monitored events. In addition, a layer-based taxonomy of adaptation-related events and a meta-model describing the dependencies among components of a cross-layer system are introduced to pinpoint the need for such a type of framework. The main strengths of this approach are the ability to handle both functional and non-functional service properties and the use of a reasoning tool deriving appropriate adaptation strategies by exploiting a set of rules. Furthermore, it has also proactive adaptation capabilities using pattern detection techniques and can be easily distributed into many computer nodes. In Section 2 a comparison table clearly states the advantages of this work.

The rest of the paper is structured as follows. Section 2 summarizes the related work. Section 3 introduces the proposed taxonomy of adaptation-related events. Section 4 presents the dependency model. Section 5 presents the proposed framework, while Section 6 exemplifies how this framework supports the case study. Finally, some concluding remarks and future work directions are presented in Section 7.

2 Related Work

The need for monitoring different functional and non-functional requirements, as well as taking adaptation actions is widely recognized by industry and academia, as a means of improving SBAs. There are several works that propose monitoring and adaptation architectures but most of them only focus on the SCC layer.

Baresi et al. [4] present an approach for self-healing of BPEL processes. This approach is based on Dynamo [3] monitoring framework along with an AOP extension of ActiveBPEL and a monitoring and recovery subsystem using Drools

ECA rules. A composition designer provides assertions for invoking, receiving or picking activities in the business process. These assertions can be specified using two domain specific languages (WSCoL and WSReL). The problem of selecting alternative services and dealing with possible interface mismatches when forwarding a request to an alternative endpoint recovery is not explicitly addressed. Additionally, the recovery rules cannot be changed dynamically, as they need to be compiled off line.

The VieDAME environment [19] extends the ActiveBPEL engine to enable BPEL process monitoring and partner service substitution based on various strategies. The services are selected according to defined selectors. VieDAME requires service registration in a repository and marking services to be monitored and eventually substituted as replaceable. It uses an engine adapter to extend the engine's functionality, but does not explicitly address fault handling.

Barbon et al. [2] present an event-based monitoring approach, developed within the Astro project, which also extends the ActiveBPEL engine and defines RTML, an executable monitoring language to specify SBA properties. Events are combined by exploiting past-time temporal logics and statistical functions. Monitors run in parallel with the BPEL process as independent software modules verifying the guarantee terms by intercepting the input or output messages received or sent by the process. This work does not allow for dynamic (re-)configuration of the monitoring system in terms of rules and meta-level parameters.

In [24,17] the authors present an approach towards extending WS-Agreement [1]. This approach supports monitoring of functional and non-functional properties. EC-Assertion is introduced to specify service guarantees in terms of different types of events, which is defined by a separate XML schema. It is based on Event Calculus (EC) [23]. By proceeding in parallel with the business process execution, it leads to less impact on performance but also to a smaller degree of responsiveness in discovering erroneous situations.

Farrel et al. [8] present an SLA-based monitoring approach exploiting EC as the underlying formalism. This approach addresses the utility computing domain, where the cornerstone aspect is to provide resources with certain quality characteristics. Contracts are defined based on contract patterns, which are then axiomatized using EC. Then, the effects of critical events on the contract state/evolution are defined. The respective framework is based on a particular architecture and comprises an analyzer managing the contract analysis and reporting, and a visualizer representing the SLA monitoring results.

A platform for developing, deploying, and executing SBAs is proposed in [6], incorporating tools and facilities for checking, monitoring, and enforcing service requirements expressed in WS-Policy notations. The Colombo platform comes with a module that manages policy assertions. Apart from evaluating the assertions attached to particular service-related entities at both design and runtime, the framework provides the means for policy enforcement, e.g., it may approve a message's delivery of a message, reject it, or defer further processing.

Despite the previous layer-specific approaches, some approaches towards cross-layer service monitoring and adaptation have been proposed. Gjørven et al. [9]

propose a coarse-grained approach, which exploits mechanisms across two layers (Service Interface and Application corresponding to ours SCC and SI layers) in a coordinated fashion. Kazhamiakin et al. [12] define appropriate mechanisms and techniques to address the adaptation requirements and constitute an integrated cross-layer framework. Both approaches tackle the problem in an inflexible manner, as the adaptation logic is predefined and static. Popescu et al. [22] provide support for dynamic cross-layer adaptation using adaptation templates, composed either directly, through invocations of WSDL operations or indirectly, through events. This approach, though, does not consider the Infrastructure layer. Finally, Zengin et al. [26] introduce an adaptation manager (CLAM) that can deal with cross- and multi-layer adaptation problems. This approach ranks a set of adaptation paths, after constructing and adaptation tree starting form an initial adaptation trigger at any of the three SBA layers.

A summary of all the aforementioned approaches is presented in Table 1. The approaches are compared according to their (cross-layer) monitoring and adaptation capabilities, dynamicity, intrusiveness and timeliness. These properties acquire a different meaning towards monitoring and adaptation. The dynamicity of a monitoring framework concerns the ability of the framework to change monitoring properties during the execution of the process whereas the dynamicity of an adaptation framework allows additions and deletions of adaptation rules. Moreover approaches which perform monitoring by weaving code that implements the required checks inside the code of the system that is being monitored are concerned as intrusive approaches. Regarding adaptation approaches, a framework is intrusive whether the applied adaptation actions change the process. We assume that intrusiveness is not desirable for monitoring unlike adaptation. The timeliness of monitoring system presents the ability of the framework to signal violations of the monitoring properties the time they occur and not after the termination of the instance. On the other hand timeliness of an adaptation framework is about the proactive or the reactive execution of the adaptation actions. Furthermore the kind and the scope of the monitored information is provided. The former one refer to the functional and non functional properties of a SBA while the latter one to instance or class application of the approach. Finally, the last two properties refer to the availability of a dependency meta-model describing the dependencies among the SBA layers and a taxonomy of monitored events.

As shown in Table 1 few of current monitoring and adaptation approaches deal with cross-layer issues [9,22,26]. These works focus mainly on adaptation process and lack some important properties that ECMAF efficiently address, such as proactive adaptation, functional and non-functional aspects consideration and wide scope, enriched with a dependency meta-model and an event taxonomy.

3 Taxonomy of Adaptation-Related Monitored Events

Many of the proposed monitoring approaches can detect different event types. These events deliver information about the SBA evolution and context change.

Table 1. Comparison Table

Approach Name	Monitoring & Adaptation Framework	Cross-Layer	Dynamicity		Intrusiveness		Timeliness		Type of Properties		Dependency Meta-Model	Event Taxonomy
			Mon.	Adapt.	Mon.	Adapt.	Mon.	Adapt.	Kind	Scope		
Dynamo [4]	Yes	No	Yes	No	Yes	Yes	Blocking pre- and post condition	Reactive	Functional Non- Functional	Instance	No	No
Viedame [19]	Yes	No	No	Yes	No	No	As soon as violations occur	Reactive	Non-Functional	Instance	No	No
Astro [2]	No	No	No	-	No	-	Signal information of interest as soon as they occur	-	Functional Non-Functional	Instance-Class	No	No
Everest [17,24]	No	No	No	-	No	-	Post-mortem	-	Functional Non-Functional	Instance	No	No
ECSTA [8]	No	No	No	-	No	-	As soon as violations occur	-	Non-Functional	Instance	No	No
Colombo [6]	No	No	No	-	Yes	-	Before a message leaves the system, or before the incoming message is processed.	-	Functional Non-Functional	Instance	No	No
QUA [9]	No	Yes	-	Yes	-	No	-	Reactive	Functional	Instance	No	No
PSLBC [22]	Yes	Yes	No	Yes	No	Yes	As soon as violations occur	Reactive	Functional	Instance	No	Yes
CLAM [26]	Yes	Yes	No	Yes	No	Yes	As soon as violations occur	Reactive	Non-Functional	Instance	Yes	No
ECMAF	Yes	Yes	Yes	Yes	No	Yes	As soon as violations occur	Proactive	Functional Non- Functional	Instance-Class	Yes	Yes

They are used to indicate whether the SBA execution evolves normally and whether there are some deviations or even violations of the desired or expected functionality. Most events are recurring during service executions and usually with the same order. Thus, it is desirable to introduce a taxonomy of monitored events to enable the mapping between these events and the suitable adaptation strategies as well as the event derivation applied in the proposed framework.

The taxonomies of common monitored events proposed are either generic or domain-specific (e.g. real-time SBAs). [22] introduces an event taxonomy for three possible application layers: organization, behavior and service, to semi-automate the discovery and selection of adaptation templates needed to fulfil complex adaptation requirements. [14] categorizes monitored events into Interface-level mismatches, i.e., services with similar functionality but through different WSDL interfaces, and Protocol-level mismatches, i.e., mismatches concerning the order or number of supplied and required messages.

The proposed high-level event taxonomy is based on two different criteria: a) the affected SBA layer and b) the service aspect concerned (functional, non-functional). The affected layer concerns the three functional layers analyzed in Section 1, while the service aspect concerns the service functional and non-functional characteristics [20]. The former ones detail the operational aspects that define the overall service behavior, such as the way and time it is invoked, while the latter concern quality attributes (e.g., response time and throughput). Its main advantage upon the other taxonomies is that it fits perfectly to the adopted SBA layers as well as the consideration of both functional and non-functional service aspects.

Fig. 1 illustrates the proposed taxonomy of adaptation-related events for the three functional SBA layers. Indicatively, at the BPM layer there are classified mismatches regarding the business process, such as KPI violations or monitored events stemming from modifications at this layer (e.g., business goal or process model modifications). At the SCC layer, monitored events focus on mismatches about service execution and QoS violations, such as I/O failures and SLA violations. Finally, at the SI layer, events mainly concern device failures affecting the overall SBA, such as limited resources or network failures.

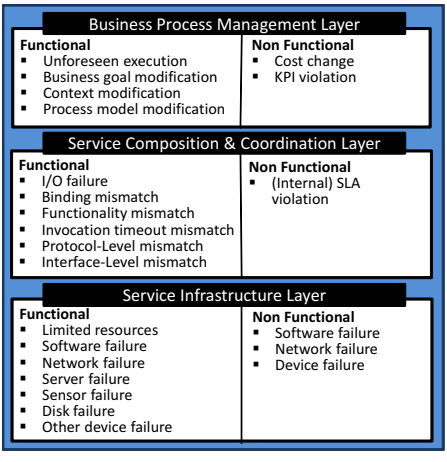


Fig. 1. Taxonomy of adaptation-related monitored events

4 Dependency Meta-model

When faults occur, it is imperative to be able to detect the root of the problem by following a process, usually called root-cause analysis. Such a process benefits from the use of a specific model called *dependency model*. This model should describe both the functional and non-functional dependencies between the system components across all possible layers to enable the detection of the component causing the fault through a top-down traversal of the respective dependencies, starting from the component where the problem is detected. Dependency models should also allow a bottom-up traversal of the respective dependencies. Such a traversal is essential so as to enable the derivation of events at higher-layers with respect to the events occuring at lower levels. Moreover, dependency models should be able to describe both static as well as dynamic component dependencies. The former are usually known and established at design time, while the latter are detected and created at runtime. Dependencies may also change during the lifetime of a SBA or system. For example, the memory requirements for installing a service may be different from those related to the service execution.

Dependency models must also conform to a particular structure and must be described in a formal way so as to enable reasoning on them. To this end, a

novel dependency meta-model is proposed in this paper, which has been specified through the OWL ontology and is depicted in Fig. 2. This meta-model has been carefully designed to capture all the relevant aspects of component dependencies.

The central concept in this meta-model is *Dependency*, which is characterized by the following four main properties [13]: a) *strength*: how strong (optional or mandatory) is the dependency between two or more components, b) *formalization*: what is the dependency's degree of formalization which directly relates to the automation degree with respect to the dependency's capturing, c) *criticality*: how critical is to capture this dependency, d) and *period*: what is the time period for which this dependency holds. Dependencies can be either *Functional* or *NonFunctional*. Functional dependencies exist between functional components, while non-functional dependencies typically exist between non-functional components. Both the *FunctionalComponent* and *NonFunctionalComponent* concept are sub-concepts of *Component*. Functional components are characterized by two main properties: a) *type*: indicating if the component is a hardware, software, or logical entity (e.g. service, activity), and b) *activity*: if the component can be directly queried or instrumented or requires the existence of an intermediary for obtaining the component's information. On the other hand, a non-functional component can be either a *QoSAttribute* or *QoSMetric*.

Non-functional dependencies can be *Qualitative* or *Quantitative*. Both dependency types can be expressed with the OWL-Q [16] semantic, non-functional based service description language, which has been slightly extended to enable the description of process and infrastructural quality attributes and metrics. Fig. 2 shows with black color which novel non-functional concepts have been introduced and with grey color which were the original OWL-Q concepts. Qualitative dependencies between two non-functional components describe particular, general non-functional relationships which can be either only qualitatively assessed, or also quantitatively assessed through e.g. instrumentation but their quantitative dependency model holds only for particular systems and services. Such a type of dependencies is characterized by two main properties: a) *valueDirection*: indicating that the value of the first non-functional component changes in the same or opposite way with respect to the value of the second one; b) *valueImpact*: describes the impact that the first non-functional component's value has on the second non-functional component's value.

The OWL-Q extensions introduced allow the description of quantitative dependencies across the same or different layers. In particular, three different metric types have been introduced: *ProcessMetric*s, *ServiceMetric*s, and *InfrMetric*s, where each metric type not only corresponds to one of the respective layers considered but also measures a particular *QoSAttribute* and applies to particular functional components at this layer. Similarly, a *QoSAttribute* can be either a *ProcessAttribute*, a *ServiceAttribute*, or an *InfrAttribute*. Each metric can then be measured through the application of a *Function* on other metrics at the same or lower layers. For example, a *ProcessMetric* can be measured through process, service, and infrastructural metrics and concerns a particular process component, such as the process itself or one of its activities. It must be noted that OWL-Q

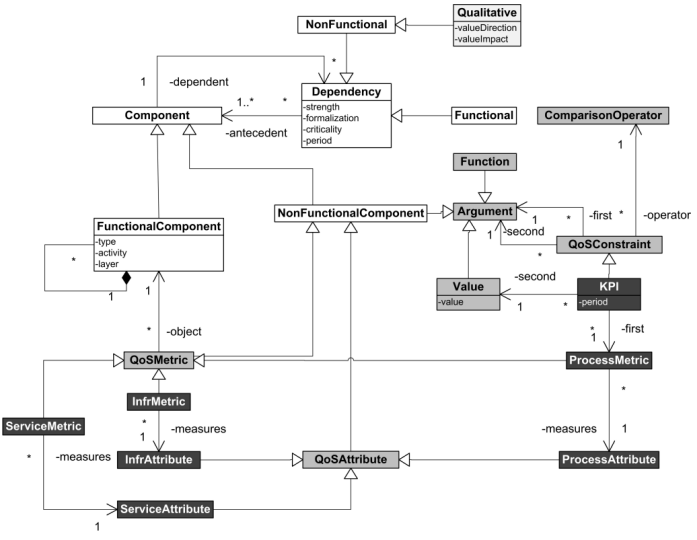


Fig. 2. The dependency meta-model

is already capable of describing the way both single and composite metrics can be computed from measurement directives and other metrics, respectively.

Therefore, OWL-Q allows for the description of both quantitative and qualitative cross-layer metric models. Quantitative models can be exploited not only for monitoring but also for conformance checking as they do not only allow the calculation of the values of the metrics at the higher levels from the values of metrics at the same or lower levels, but also the comparison of the computed values against the stated requirements. On the other hand, qualitative models allow inspecting the correctness of the monitored values, as e.g. particular qualitative dependencies may not be respected by the monitored values of specific quality components, produced by particular error-prone sources of information.

5 Monitoring and Adaptation Framework

Fig. 3 presents the architecture of the proposed cross-layer monitoring and adaptation framework. This framework comprises a Monitoring Engine able to collect the monitored events during the service execution, an Adaptation Engine able to perform adaptation actions, and an Execution Engine. The first two engines communicate with each other via events through a publish/subscribe mechanism.

Monitoring Engine. The Monitoring Engine comprises a **Monitor Manager** and a number of individual **Monitoring Components**. Each of the latter components is assigned to detect events at a specific SBA layer and immediately deliver them to the Monitor Manager. The Monitor Manager, in turn, continuously delivers information about the service execution produced by the Execution Engine while collecting events from the Monitoring Components. The Monitor

Manager communicates with the Translator via a publish/subscribe mechanism. A needed monitored event is delivered to the Translator as soon as it is detected. It is imperative to send the events in the order that they are received so as to have an as reliable as possible pattern matching mechanism. As there are many Monitoring Components delivering events, specific techniques are required to ensure this, such as event timing [18] and clock synchronization [15].

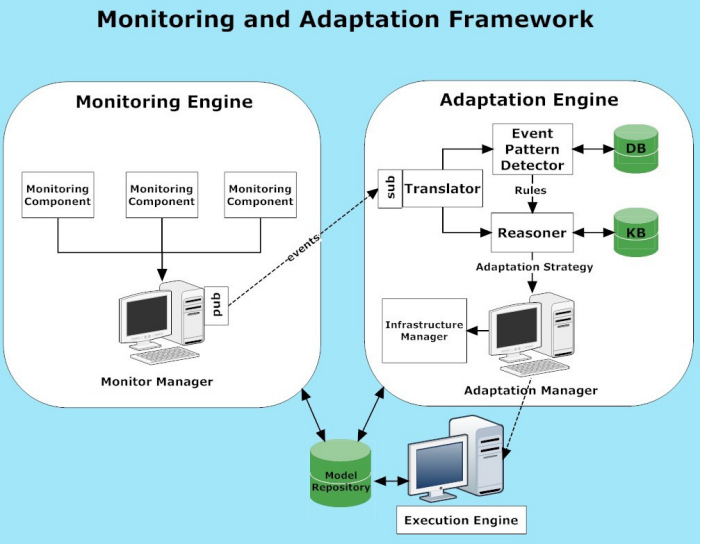


Fig. 3. Architecture of Monitoring and Adaptation framework

Adaptation Engine. The Adaptation Engine comprises a number of components supported by suitable repositories:

- The **Translator**, also incorporating a *subscribe mechanism*, receives the events sent by the Monitor Manager and translates them into a suitable format required by the Event Pattern Detector and Reasoner components.
- It is very common that failures at the SI layer lead to other failures and violations both at the same and higher layers, forming a chain of monitored events. So, it is desirable to detect those event patterns causing service failures by exploiting specific mechanisms (e.g. pattern matching ones [11]). These patterns are detected by the **Event Pattern Detector**.
  Our approach relies on using such a technique to detect dynamically patterns of monitored events to prevent other chains of events from occurring. As such, it enables the *proactive service adaptation* by mapping the detected patterns to suitable adaptation strategies. For example, having received two monitored events the Event Pattern Detector detects a specific pattern composed of one more event. This pattern is mapped to an adaptation strategy, and then, this mapping is translated into a rule that is passed to the Reasoner.

The adaptation strategy which is finally derived from the Reasoner, into an appropriate format, prevents the third event from occurring. A **Database** supplies the needed information for the mapping and the translation.

- The **Reasoner** receives events from the Translator and rules from the Event Pattern Detector, and derives an adaptation strategy to be performed by the **Adaptation Manager**. The rules provide the required data to perform the derivation, supported by a Knowledge Base (KB), containing the appropriate information to take the right decision.
- The **Adaptation Manager** executes the adaptation strategy exported by the Reasoner with the aid of two components. The **Infrastructure Manager** is able to treat malfunctions regarding the SI layer, which is the main source of many service failures, and the **Model Repository** supplies the appropriate information, such as service descriptions and requirements, layer dependencies, and metric and SLA models, so as to fulfil the supported adaptation strategies. The Execution Engine is called to support the adaptation process, especially for strategies regarding the BPM and SCC layers.

The main benefits of the monitoring and adaptation framework are as follows:

- **Distributed workload**. As there are layer-specific Monitoring Components that pass monitored events to the Monitor Manager, monitoring is distributed among the available monitoring mechanisms. In addition, there can be many computer nodes with a separate Monitoring and Infrastructure Manager component, that can a handle a portion of the whole monitoring and adaptation workload, as depicted in Fig. 4.
- **Extensibility**. As services evolve, new monitoring and adaptation techniques are required in order to cope with continuous context changes and other unpredictable malfunctions. This framework can integrate such techniques with the existing ones, while preserving its functionality and integrity.
- **Cross-layer capability**. The framework is able to support all three SBA functional layers. It incorporates mechanisms to detect events across all layers and derive additional events using pattern matching techniques.
- **Pro-active adaptation**. The use of pattern matching techniques, as well as the mapping between patterns and adaptation strategies allows for proactively adapting the SBA.

Some preliminary implementation solutions have already been investigated. As far as the monitoring is concerned we plan to use the Astro project [2], which has already been discussed in Section 2. In addition, we have decided to use the ESPER language (`http://esper.codehaus.org`) for the specification of the events and the pattern detection process. ESPER provide efficient event processing, comprehensive pattern detection and publish/subscribe capabilities. Finally, the reasoning process can be efficiently accomplished by the powerful EC-Jess Reasoner [21], which translates Event Calculus axioms into Jess rules and then encodes the domain description as a Jess program. Its main benefits are that it handles Event Calculus rules and produces models as output. The joining of these mechanisms as well as other gaps filling within this framework is in our future plans.

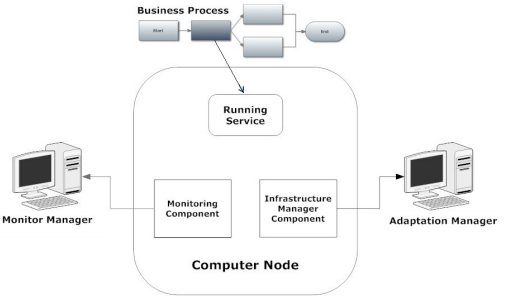


Fig. 4. Distributed workload performed by a computer node

6 Traffic Management Case Study

This case study describes a traffic management system designed to manage normal traffic situations as well as emergency cases [7]. Such emergency case handling includes several different actions, such as directing the rescue forces to the accident location and managing traffic deviations. Fig. 6 and Fig. 5 respectively illustrate these two cases. Each figure depicts the three functional layers. In both cases, workflow tasks are executed either manually or by mapping them on (Web) services. Each service is then mapped to the appropriate infrastructure.

The actors involved are *traffic managers*, i.e., individuals accountable for entities controlling the traffic management system, generic *rescue forces* (e.g., police and ambulances), and *citizens*, such as motorists and pedestrians.

Fig. 6 illustrates normal traffic conditions, where the system tries to optimize some parameters such as total noise, overall throughput, and air pollution. In particular, the system shall consider different needs, such as the ones of pedestrians and motorists, and other factors like heavy traffic, public events, school and working hours, holidays or public regulations which may alter traffic demand and needs during conditions that do not involve emergencies. The system interrupts the Normal Traffic Situation process, when an accident happens, and jumps to the Critical Traffic Situation subprocess.

Fig. 5 depicts a critical traffic situation, in which a serious car accident occurs at a central road. In particular, the involved citizens inform the traffic manager that must control the overall traffic situation (control traffic devices, inform citizens) and assess the incident so as to inform the appropriate rescue forces about the accident and direct them to the specific location. Moreover, the traffic manager monitors the environment variables, such as air pollution and noise. Different adaptation actions must be taken by the traffic manager as well as by the rescue forces, such as:

- Traffic management device reconfiguration (e.g., traffic lights) by the traffic manager, in order to reduce stop-and-go traffic. This should also help to keep air pollution low, even if it is not critical during emergency situations.
- Accident reporting to citizens via their devices (e.g, GPS, mobile phones) by the traffic manager to avoid traffic congestion at the accident location.

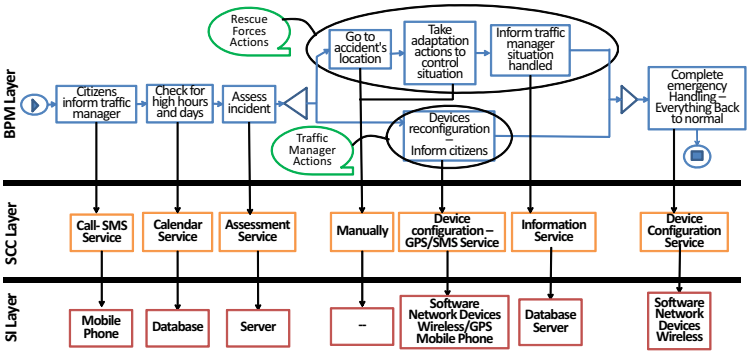


Fig. 5. Critical traffic situation

– Traffic closing/limiting to or from the involved location by the rescue forces.
– Traffic deviation by the rescue forces through alternative places not intended for heavy traffic.

After a complete emergency handling, there is a gradual return back to the normal situation. The rescue forces inform the traffic manager, who updates the system and informs the citizens through their devices.

As already discussed, there are various dependencies among the three SBA layers. The occurrence of a failure at one layer may result in a failure at other layers. This work aims at locating the failure event and taking adaptation actions in order to prevent its spread at the others layers, as soon as possible.

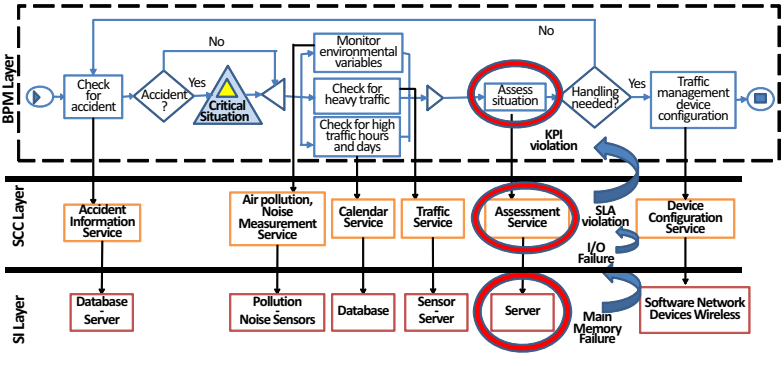


Fig. 6. Layers' interaction during normal traffic conditions

Fig. 6 presents an illustrative example. Suppose that a KPI exists dictating that the maximum duration of the process should be less than 10 seconds. Further, suppose that an SLA exists for the assessment service *AS* dictating that its maximum execution time must be less than 6 seconds. Thus, as it can be seen,

a violation of the respective SLA constraint may cause a violation to the KPI, by considering that the previous process activities do not run longer than 3 seconds. It must also be indicated that *AS*'s execution time is inverse proportional to the main memory size and the CPU percentage allocated for its execution. Moreover, there is a low limit for the main memory allocated, after which the SLA violation will be unavoidable as the service behavior will be unpredictable and even if it does not fail it will certainly take a long time to execute. In fact, after 2 seconds from the *AS*'s execution, the main memory allocated to it has indeed surpassed the low level of 50 MB.

A Monitoring Component, running at the server where *AS* is executed, detects that the available main memory is not sufficient (SI layer) for *AS*. At the same time, another Monitoring Component detects that there is an I/O failure at the SCC layer as *AS* has produced a wrong output. Both events are first sent to the Translator, which transforms them to the appropriate format and sends them to the Reasoner. Based on the two events received, a specific rule is fired which derives that the best strategy is to execute another instance of the *AS* service at a more powerful server and with a better memory and CPU allocation. The suitability of the strategy lies on the fact that by executing a "better" service instance and with better allocation for the hardware resources, the probability that the SLA is not violated becomes very high (as we do not know if another failure may occur in the near future regarding the new instance) and in this way also the KPI violation may be avoided. Such a rule has been derived by the Event Pattern Detector based on the previous history log and has been already inserted into the Reasoner. The derived strategy is sent to the Adaptation Manager which executes it with the assistance of the Infrastructure Manager and the Execution Engine.

As can be understood, ECMAF handles perfectly such a cross-layer scenario. The adaptation actions performed are the appropriate ones, based on the event history and the current context. Moreover, the dependencies among the layers are clearly discerned.

7 Conclusions and Future Work

To sum up, this paper describes a framework that is able to detect monitored events across the three functional layers of a SBA and derive suitable adaptation strategies through a reasoning mechanism. The communication between the monitoring and adaptation engines is achieved by a publish/subscribe mechanism. In addition to the framework's description, a taxonomy of adaptation-related events and a dependency meta-model between the system components across all functional SBA layers are provided, in order to pinpoint the need for such a cross-layer approach. The main benefits of this approach are that it can handle both functional and non-functional service aspects, as well as it can proactively adapt the SBA across all the functional layers.

The following future directions are planned. First, developing such a distributed cross-layer monitoring and adaptation framework, using existing technologies and mechanisms and extending them if necessary. Second, extending

the current taxonomy of adaptation-related events, exploiting additional aspects. Third, extending the proposed dependency meta-model. Finally, experimentally evaluating and optimizing the framework.

Acknowledgements. 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).

References

1. Andrieux, A., et al.: Web Services Agreement Specification (March 2007), http://forge.gridforum.org/sf/docman/do/downloadDocument/projects.graap-wg/docman.root.published_documents.web_services_agreement_specifica/doc14574
2. Barbon, F., Traverso, P., Pistore, M., Trainotti, M.: Run-time Monitoring of Instances and Classes of Web Service Compositions. In: ICWS, pp. 63–71. IEEE (2006)
3. Baresi, L., Guinea, S.: Dynamo: Dynamic Monitoring of WS-BPEL Processes. In: Benatallah, B., Casati, F., Traverso, P. (eds.) ICSOC 2005. LNCS, vol. 3826, pp. 478–483. Springer, Heidelberg (2005)
4. Baresi, L., Guinea, S., Pasquale, L.: Self-healing BPEL Processes with Dynamo and the JBoss Rule Engine. In: ESSPE 2007 in conjunction with the 6th ESEC/FSE joint meeting, pp. 11–20. ACM (2007)
5. Benbernou, S., Cavallaro, L., Hacid, M.S., Kazhamiakin, R., Kecskemeti, G., Pazat, J.L., Silvestri, F., Uhlig, M., Wetzstein, B.: PO-JRA-1.2.1, State of the Art Report, Gap Analysis of Knowledge on Principles, Techniques and Methodologies for Monitoring and Adaptation of SBAs. Tech. rep., S-cube (July 2008)
6. Curbera, F., Duftler, M., Khalaf, R., Nagy, W., Mukhi, N., Weerawarana, S.: Colombo: Lightweight middleware for service-oriented computing. IBM Systems Journal 44(4), 799–820 (2005)
7. Di Nitto, E., Mazza, V., Mocci, A.: Collection of industrial best practices, scenarios and business cases (2009)
8. Farrell, A., Sergot, M., Salle, M., Bartolini, C.: Using the Event Calculus for the Performance Monitoring of Service-Level Agreements for Utility Computing. In: WEC, vol. 6. Citeseer (2004)
9. Gjørven, E., Rouvoy, R., Eliassen, F.: Cross-layer self-adaptation of service-oriented architectures. In: MW4SOC, pp. 37–42. ACM (2008)
10. Hielscher, J., Kazhamiakin, R., Metzger, A., Pistore, M.: A Framework for Proactive Self-adaptation of Service-Based Applications Based on Online Testing. In: Mähönen, P., Pohl, K., Priol, T. (eds.) ServiceWave 2008. LNCS, vol. 5377, pp. 122–133. Springer, Heidelberg (2008)
11. Karp, R.M., Rabin, M.: Efficient randomized pattern-matching algorithms. IBM Journal Research and Development 31(2), 249–260 (1987)
12. Kazhamiakin, R., Pistore, M., Zengin, A.: Cross-Layer Adaptation and Monitoring of Service-Based Applications. In: Dan, A., Gittler, F., Toumani, F. (eds.) ICSOC/ServiceWave 2009. LNCS, vol. 6275, pp. 325–334. Springer, Heidelberg (2010)
13. Keller, A., Blumenthal, U., Kar, G.: Classification and Computation of Dependencies for Distributed Management. In: ISCC. IEEE, Antibes (2000)

14. Kongdenfha, W., Motahari-Nezhad, H.R., Benatallah, B., Casati, F., Saint-Paul, R.: Mismatch Patterns and Adaptation Aspects: A Foundation for Rapid Development of Web Service Adapters. IEEE Trans. Serv. Comput. 2, 94–107 (2009)
15. Kopetz, H., Ochsenreiter, W.: Clock synchronization in distributed real-time systems. IEEE Trans. Computers 36(8), 933–940 (1987), http://dblp.uni-trier.de/db/journals/tc/tc36.html#KopetzO87
16. Kritikos, K., Plexousakis, D.: Semantic QoS Metric Matching. In: ECOWS. IEEE Computer Society, Zurich (2006)
17. Mahbub, K., Spanoudakis, G.: Monitoring WS-Agreements: An Event Calculus-Based Approach. Springer (2007)
18. Mok, A.K., Liu, G.: Efficient run-time monitoring of timing constraints. In: IEEE Real-Time and Embedded Technology and Applications Symposium, p. 252 (1997)
19. Moser, O., Rosenberg, F., Dustdar, S.: Non-intrusive Monitoring and Service Adaptation for WS-BPEL. In: WWW, pp. 815–824. ACM (2008)
20. Papazoglou, M.P.: Web Services: Principles and Technology. Pearson, Prentice Hall (2008)
21. Patkos, T., Plexousakis, D.: DECKT: Epistemic Reasoning for Ambient Intelligence. ERCIM News (84), 30–31 (2011)
22. Popescu, R., Staikopoulos, A., Liu, P., Brogi, A., Clarke, S.: Taxonomy-driven Adaptation of Multi-Layer Applications using Templates. In: SASO (October 2010)
23. Shanahan, M.: The Event Calculus Explained. In: Veloso, M.M., Wooldridge, M.J. (eds.) Artificial Intelligence Today. LNCS (LNAI), vol. 1600, pp. 409–430. Springer, Heidelberg (1999)
24. Spanoudakis, G., Mahbub, K.: Non-Intrusive Monitoring of Service-Based Systems. International Journal of Cooperative Information Systems 15(3), 325–358 (2006)
25. Zeginis, C.: Monitoring the QoS of Web Services using SLAs - Computing metrics for composed services. Master's thesis, University of Crete, Greece (2009), http://www.csd.uoc.gr/~zegchris/master_thesis.pdf
26. Zengin, A., Marconi, A., Pistore, M.: CLAM: Cross-layer Adaptation Manager for Service-Based Applications. In: QASBA 2011, pp. 21–27. ACM (2011)

Phd Symposium Preface

Quan Z. Sheng[1], Cesare Pautasso[2], and Sonia Ben Mokhtar[3]

[1] The University of Adelaide, Australia
[2] The University of Lugano, Switzerland
[3] LIRIS, CNRS, France
qsheng@cs.adelaide.edu.au, Cesare.pautasso@usi.ch,
sonia.ben-mokhtar@liris.cnrs.fr

Service oriented computing (SOC) has rapidly transformed from a vision, in the beginning of the century, to reality with technologies such as Web services, Cloud services, and the Internet of Things. While this has provided the industry and practitioners with the opportunities for a new generation of products and services, it has also raised many fundamental research challenges and open issues. The International Conference on Service Oriented Computing (ICSOC) is a premier annual event for researchers, practitioners and industry leaders to discuss and share the success and achievements in this vibrant and rapidly expanding area.

The ICSOC PhD Symposium, as part of the ICSOC conference, is an international forum for PhD students working in the broad areas of service computing, Web services and service engineering to present and discuss emerging research problems and ideas on how to tackle these issues. The goals of the ICSOC PhD Symposium event are:

- To bring together PhD students and established researchers in the field of service oriented computing and related areas,
- To enable PhD students to interact with other PhD students and to stimulate an exchange of ideas, suggestions, and experiences among participants,
- To give PhD students the opportunity to present and discuss their research in a constructive and critical atmosphere,
- To provide students with fruitful feedback and advice on their research and thesis proposals.

To achieve these goals, the symposium operates in a workshop format, giving PhD students an opportunity to showcase their research and providing them with ample feedback from senior international researchers and peer PhD students. This year, for the first time, we successfully experimented with a pair-presentation format, where PhD students introduced one another and had to prepare questions for each other in advance.

The symposium in Paphos, Cyprus is the 7th PhD Symposium of a series held without interruption in conjunction with the ICSOC conferences since 2005. This year we received 18 submissions from 10 countries. Each paper has been reviewed by at least three members of the Program Committee. The submissions were evaluated

G. Pallis et al. (Eds.): ICSOC 2011, LNCS 7221, pp. 162–163, 2012.

based on originality (novelty), problem significance, technical/scientific quality, and presentation/style. Each reviewer also provided detailed suggestions intended for improving the research work of the students. The Program Committee eventually selected 12 papers for the presentation and discussion in the symposium. The selected papers cover a wide range of topics in SOC from service engineering, quality of service, privacy and trust to service modelling, service composition, as well as emerging areas such as Web of Things and Cloud computing.

We thank all authors for their submissions and their active participation and the Program Committee members for their excellent work. We would also like to thank IBM Research, USA that has generously offered financial support to the symposium. We specifically thank Francisco Curbera from IBM Research for his continuous support of this event.

December 2011

Quan Z. Sheng
Cesare Pautasso
Sonia Ben Mokhtar

Program Committee

Boualem Benatallah	University of New South Wales, Australia
Djamal Benslimane	University of Lyon, France
Walter Binder	University of Lugano, Switzerland
Athman Bouguettaya	CSIRO ICT Centre, Australia
Mauro Caporuscio	Politecnico di Milano, Italy
Florian Daniel	University of Trento, Italy
Marlon Dumas	University of Tartu, Estonia
Nikolaos Georgantas	INRIA Paris, France
Mohand-Said Hacid	LIRIS, France
Xitong Li	MIT, USA
Kwei-Jay Lin	University of California, Irvine, USA
Pascal Poizat	University of Evry Val d'Essonne, France
Aviv Segev	KAIST, Korea
Jian Yu	Swinburne University of Technology, Australia

Defining an SLA-Aware Method to Test Service-Oriented Systems

Marcos Palacios
Supervised by: José García-Fanjul and Javier Tuya

University of Oviedo
palacios@lsi.uniovi.es

Abstract. In the scope of Service Oriented Architectures (SOA), Service Level Agreements (SLAs) are used to specify the conditions that have to be fulfilled by both service provider and consumer. These stakeholders need checking whether the executions of the services fulfill the conditions or not, so the evaluation is an important and not trivial task within the testing process. This paper describes the current state of an ongoing PhD research that aims to define an SLA-aware testing method to test service-oriented systems. A model will represent relevant information associated to the terms of the SLA and, from this model, interesting situations will be identified and prioritized. The exercitation of these situations will be performed through a suited design and execution of test cases. In addition to this, the approach could be complemented with monitoring techniques, determining which situations have already been executed in an operation environment in order to select a set of situations that remain unexercised and need to be tested.

Keywords: Service Level Agreements, Software Testing, Monitoring, Service Oriented Architectures.

1 Scope and Motivation

During last decade Service Oriented Architecture (SOA) has emerged as a promising solution to develop interoperable and highly dynamic applications by integrating available services over the web. In some scenarios it is necessary that the service provider and the consumer negotiate and agree a set of conditions related to the use of the services. Such conditions are often specified in a contract or technical document named Service Level Agreement (SLA).

Within the tasks included in the management of the SLAs, testing has been identified as a challenge. Monitoring is often proposed as a suited technique to observe the behaviour of the software and detect whether the services are violating the conditions of the SLA but these approaches have limitations. First of all, the problems in the software are detected after they have occurred (reactive approaches) so consequences can not be avoided. Furthermore, there can be situations that are not exercised during the period of time the system is being monitored but they may occur in the future. In critical scenarios where an SLA violation may lead to severe consequences for stakeholders, monitoring could be complemented with testing methods that anticipate the detection of problems identifying and exercising interesting situations through a suited test design (proactive

G. Pallis et al. (Eds.): ICSOC 2011, LNCS 7221, pp. 164–170, 2012.

approaches). This research will mainly focus on defining an SLA-aware method that allows testing systems in the aforementioned scenarios.

The content of the paper is organized as follows. Section 2 outlines the research focus and the objectives of this PhD research. Section 3 and 4 present the ongoing work with results achieved so far and our proposal to evaluate and test SLAs. In Section 5 we briefly describe the state of the research in the addressed topic. Finally, conclusions and expected results are outlined in Section 6.

2 Research Focus

The final objective of this research is to detect faults in the software that can lead to SLA violations so their correspondent consequences can be avoided or mitigated. Hence, the main question can be defined as follows:

- ***How can we test service-oriented systems with a defined SLA before the executions cause non-expected consequences for the stakeholders?***

This goal can be achieved addressing the following set of research questions:

1. How can an SLA be evaluated for purposes of detecting faults in the software?
2. How can we represent the content of the SLA and other relevant objective information in a model?
3. What potential error-prone situations can be identified from such model?
4. How can these situations be exercised through a test design and execution?

This ongoing PhD research aims at contributing to solve the problem of testing service-oriented systems with the elaboration of a test method that addresses the aforementioned questions. Furthermore, we will also consider complementarity between testing and monitoring:

5. How can we exploit the benefits of monitoring to collect information when the software is deployed in an operation environment?

3 SLAs and Their Evaluation

During the first steps of this ongoing research, we studied the state of the art in the field of testing service-oriented systems that allow a specific capability such as dynamic binding [8]. In this work, we identified that the testing of SLAs is a promising research line that has not probably received enough effort yet. Furthermore, although different languages have been proposed with the aim at standardizing the specification of SLAs, it has been WS-Agreement (WSAG) [1] which has received more attention so far. The specification of a WSAG is composed of three parts: name, context and terms. The latter describe the obligations that have to be fulfilled and are the most important elements of the SLA. A guarantee term includes the *Scope* which is the list of services this guarantee applies to, a *Qualifying Condition* which is an assertion under which the term is applied, and a *Service Level Objective* (SLO) that has to be guaranteed. These terms can be combined using three compositor elements: *All*, *OneOrMore* and *ExactlyOne* (logical AND, OR and XOR respectively).

The evaluation of an SLA requires making a decision about each of the elements specified in such SLA. Typically, the evaluation of an SLA may be depicted with a two-way traffic light indicator (green / red) which represents whether the agreement has been fulfilled or violated respectively. In order to make this decision, it is necessary to check the evaluation of all the terms of the SLA, also taking into account the compositor elements. Hence, the term can be considered as the most indivisible element of the SLA. At runtime and after having analyzed the collected information:

- A term is considered ***Fulfilled / Violated*** if and only if the services specified in the *Scope* have been executed, the *Qualifying Condition* is honored so the term is valid and the conditions specified in the *SLO* are satisfied /are not satisfied.

In addition to this two values *Fulfilled* and *Violated*, it is possible to identify and, in fact, WSAG does it, a third potential value that a term can take after its evaluation.

- A term is considered **Not Determined** if and only if the services specified in the *Scope* have not been yet executed.

The interpretation of this value according to WS-Agreement is that, at the moment of the evaluation, no activity regarding the term has happened yet or no activity is currently happening that allows evaluating whether the term is fulfilled or violated.

After studying the internal elements of a term and its interpretation according to the standard, we have identified a new situation where the term can be found after its evaluation and which has not been explicitly identified in WS-Agreement. This situation arises when the services specified in the Scope of the term have been executed but those executions do not satisfy the Qualifying Condition so such term is not valid and should not be applied for the purpose of the evaluation of the global SLA.

- A term is considered ***Inapplicable*** if and only if the services specified in the *Scope* have been executed but the *Qualifying Condition* is not satisfied.

As an SLA specifies a hierarchy of terms combined through the compositor elements, we will define a logic that allows unequivocally representing all the potential situations that may occur during the SLA evaluation process. It is necessary that the logic includes these two additional evaluations, leading to a four-valued logic where *Not Determined* and *Inapplicable* are two similar interpretations of the treatment of the null value in the three-valued logic (broadly studied in the context of DBMS and applied in the scope of software testing). The definition of this logic allows addressing the first research question of Section 2.

In addition to the potential values that terms can take during their evaluation, there are different factors that affect the evaluation and we will have to take into account in this PhD research, for example, the scope (we may have to evaluate only an specific term, a compositor element, the whole SLA, etc.) or the point in time of the evaluation (after each service execution, a specific number of executions, a temporal window, etc.).

4 Testing SLAs

Considering the logic described in Section 3 as a foundation to evaluate the SLAs, we have planned to define an SLA-aware method that contributes to improve the process of testing service based systems. This method is depicted in Fig. 1 where the SLA is always associated to a Software Under Test (SUT).

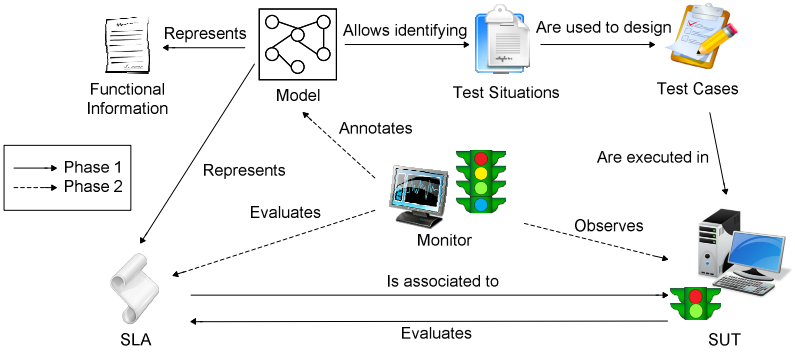


Fig. 1. SLA Test Method

One of the inherent characteristics of the management of SLAs is the capability to determine whether the conditions specified in the agreement are being fulfilled or not. Regarding this issue, the SUT must have a mechanism that observes the executions, collects data and, based on this information, performs the evaluation of the SLA. In fact, from our point of view this mechanism is part of the SUT and, hence, we assume that it also needs to be tested.

The method that will be developed during the first phase of this research is composed of the following processes, which will allow achieving our expected results:

1. Model the content of the SLA and other relevant information.
2. Define a set of criteria that allow the identification and prioritization of interesting test situations from the model.
3. Define procedures to design a set of test cases that exercise the selected situations.

Previous studies have represented the content of the SLA in a model [6][4]. In our approach, the development of this model should be useful in the process of testing service based systems and it is aligned with the second research question of Section 2. This model ought to contain, at least, all the relevant information specified in the SLA. Furthermore, we may consider representing other kind of information that could help to improve the testing process. This information can be extracted from different sources (requirements, behaviour models, etc.) and, although the SLAs typically contain non-functional characteristics, we consider that it could be interesting to represent some functional aspects of the SUT in the agreement. For instance, we are considering representing constraints about the evaluation of terms depending on the execution order of services, which could be specified using UML sequence diagrams.

From the previously constructed model and taking the proposed logic into account, different sets of test situations can be initially identified (aligned with the third research question of Section 2). The evaluation of terms, compositor elements or the global SLA with the values *Violated* (representing existing problems in the SUT), *Not Determined* and *Inapplicable* (both representing situations that have not been yet exercised) identifies potential error-prone situations that need to be tested before they lead to undesirable consequences for the stakeholders. Regarding this, a new problem may arise. In a previous work [7] we addressed the design of test specifications from

the conditions specified in the SLA. In such study, we realized that the number of situations identified from a low complexity agreement could be unmanageable. Hence, it is necessary to define criteria that allow reducing and prioritizing such situations in order to obtain a reasonable cost-effective set of tests. These criteria may be based on different factors such as business risks, economic penalty fees, etc.

The identified situations will be exercised through a suited design and execution of test cases (aligned with the fourth research question of Section 2). Typically, a test case should cover as many situations as possible in order to obtain a final reduced but effective set of test cases. This is even more important in SOA systems because we do not probably have unlimited and full access to the services but executions usually imply an economic cost. Thus, we will have to define criteria that allow us to design good test cases and decide their priority to be executed, overall if there are limitations that hinder the execution of the complete set of tests.

Finally and as we have previously outlined, we have planned to exploit the benefits of monitoring techniques to complement the approach (aligned with the fifth research question of Section 2). The observation of the software deployed in an operation environment could be a very useful task to recollect broad information and provide an interesting feedback. The monitor will show, for example, which situations remain unexercised and which situations are currently causing problems in the system. With this information, new decisions may be made in order to identify new situations or to prioritize existing ones in future regression tests.

5 Related Work

In the scope of service oriented computing, few approaches have addressed the identification of tests from the SLA specification. Di Penta et al. [3] perform black-box and white-box testing using Genetic Algorithms to detect SLA violations in service compositions where multiple services can be potentially invoked. In a previous work [7] we propose to test the conditions of the SLA specified in WSAG using a well-known testing technique, the Category Partition Method. Furthermore, there are approaches that do not specifically test SLAs but contribute to enable the necessary infrastructure. For example, Bertolino et al. [2] propose the PUPPET framework, which allows generating stubs from the WSAG, WSDL and BPEL specification of the services to test SLA-aware service compositions.

On the other hand, several works have addressed the testing of SLAs using monitoring approaches. Mahbub and Spanoudakis [6] propose to model and monitor the conditions specified in WS-Agreement using an Event Calculus (EC) based approach. Leitner et al. [5] propose a framework that allows monitoring and predicting SLA violations before they have occurred using machine learning techniques. Oriol et al. [9] aims at monitoring and detecting SLA violation at runtime using SALMon. This system allows triggering adaptive actions to correct the SLA violation after been detected. Raimondi et al. [10] proposed a system that automatically monitors SLAs, translating timeliness constraints into timed automata, which is used to verify traces of services executions.

6 Conclusions and Expected Results

This paper presents an overview of an ongoing PhD research in the field of testing service-based systems with Service Level Agreements. After having reviewed the state of the art in such research line [8], we have considered proposing an SLA test method, describing the issues we have planned to deal with, proposing potential solutions to solve them and stressing particular difficulties that arise from each of them.

In this research we expect to achieve a detailed description of how these systems can be tested according to the conditions that are specified in the SLA. This method is a proactive approach to identify, prioritize and exercise interesting test situations using a model that represents relevant information of the SLA. The second direction of the research involves exploiting the advantages of reactive approaches to improve and complete the potential information gathered from the tests.

Finally, a complementary and transversal expected result in this research will be the attempt to integrate existing solutions when possible. For instance, the execution of the tests could be performed using a framework (e.g. [2]) that makes easier the generation of a suited test infrastructure of the SUT. Furthermore, we will study how an existing monitoring system with the capability to check SLAs (e.g. [6]) could be adapted and integrated.

Acknowledgments. This PhD research is being partially funded by the Department of Science and Innovation (Spain) and ERDF funds within the National Program for Research, Development and Innovation, project Test4DBS (TIN2010-20057-C03-01) and FICYT (Government of the Principality of Asturias) Grant BP09-075.

References

1. Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Nakata, T., Pruyne, J., Rofrano, J., Tuecke, S., Xu, M.: Web Services Agreement Specification (2007)
2. 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)
3. Di Penta, M., Canfora, G., Esposito, G., Mazza, V., Bruno, M.: Search-based Testing of Service Level Agreements. In: 9th Annual Conference on Genetic and Evolutionary Computation (GECCO 2007), London, pp. 1090–1097. ACM, New York (2007)
4. Kotsokalis, C., Yahyapour, R., Rojas Gonzalez, M.A.: Modeling Service Level Agreements with Binary Decision Diagrams. In: Baresi, L., Chi, C.-H., Suzuki, J. (eds.) ICSOC-ServiceWave 2009. LNCS, vol. 5900, pp. 190–204. Springer, Heidelberg (2009)
5. Leitner, P., Michlmayr, A., Rosenberg, F., Dustdar, S.: Monitoring, Prediction and Prevention of SLA Violations in Composite Services. In: IEEE International Conference on Web Services (ICWS) Industry and Applications Track (2010)
6. Mahbub, K., Spanoudakis, G.: Monitoring WS-Agreements: an Event Calculus based Approach. Test and Analysis of Service Oriented Systems, Springer V, pp. 265–306 (2007)

7. Palacios, M., García-Fanjul, J., Tuya, J., de la Riva, C.: A Proactive Approach to Test Service Level Agreements. In: Fifth International Conference on Software Engineering Advances (ICSEA), pp. 453–458 (2010)
8. Palacios, M., García-Fanjul, J., Tuya, J.: Testing in Service Oriented Architectures with Dynamic Binding: A Mapping Study. Information and Software Technology 53(3), 171–189 (2011)
9. Oriol, M., Marco, J., Franch, X., Ameller, D.: Monitoring Adaptable SOA System using SALMon. In: Workshop of Service Monitoring, Adaptation and Beyond (MONA+), ServiceWave Conference (2008)
10. Raimondi, F., Skene, J., Emmerich, W.: Efficient Online Monitoring of Web-Service SLAs. In: Proceedings of the 16th ACM SIGSOFT Int. Symposium on Foundations of Software Engineering (SIGSOFT 2008/FSE-16) (2008)

Data Flow-Oriented Process Mining to Support Security Audits

Thomas Stocker
Supervised by: Rafael Accorsi

University of Freiburg, Germany
stocker@iig.uni-freiburg.de

Abstract. The automated execution of dynamically-evolving business processes in service-oriented architectures requires audit methods to assert that they fulfill required security properties. Process mining techniques can provide models for the actual process behavior, but mostly disregard the dynamics of processes running in highly flexible environments and neglect the data flow perspective. This research plan is on novel data-oriented mining techniques to tackle these shortcomings in order to support effective security audits.

1 Problem Statement and Context

Business process management (BPM) allows the design, enactment, management and analysis of operational business processes [19]. In such "process-aware information systems", workflow adoption is expected to soar with the advent of cloud and service-oriented computing, providing a basis for structuring and integrating business processes [8]. Valuable synergy effects between BPM and SOA gave reasons for a new field of service-oriented process modeling tools and languages such as WS-BPEL. The deployment of configurable workflows "as-a-service" [1] that may evolve along time [20] allows enterprises to work more efficient and react more flexible on changing requirements such as customer demands or changes in the technological, business or legal context.

Although workflows are largely employed in mission-critical activities demanding strong security and privacy guarantees [7], there is today a lack of audit methods to assert that they fulfill the required properties [14]. In particular, computer assisted auditing techniques (CAAT) are missing that cope with the *security analysis* of *evolving* workflows [17]. As a consequence, a significant number of exploited process vulnerabilities and data leaks [12] goes undetected. Especially in the highly dynamic field of service-oriented computing the verifiability of security guarantees plays a key role in technology-adaption and trust-establishment. Besides direct illegal data flows, information leakage includes the use of covert channels that allow data usage and flow not intended by the system, i.e. by monitoring (temporally) system behavior [13]. Information flow analysis captures both types of data flow.

This research plan is on the development of advanced process mining techniques that allow detailed views on process enactments including data flows,

G. Pallis et al. (Eds.): ICSOC 2011, LNCS 7221, pp. 171–176, 2012.

thereby providing a basis for efficient security analysis with regard to information flow policies. Process mining is a method of distilling a structured process from a set of real executions [18]. The usual target meta-model for reconstruction is (some dialect of) Petri net, which provides an expressive formalism to capture the control flow and data flow of business processes [6]. Traditionally, workflow mining extracts a single, representative model that consolidates all the different executions happening in the log file. Recently, trace clustering techniques have been proposed as a preprocessing step for workflow mining [10,15,11]. The idea is to group traces according to different characteristics and, subsequently, apply workflow mining to a particular set of clusters.

The considered problem statement is twofold and follows directly from the identified shortcomings exhibited by current trace clustering and process mining approaches:

1. **Recovered processes neglect the history of changes.**
   Traces are mostly grouped according to their structural similarity, thereby neglecting the time aspect. In doing so, auditors cannot test for particular timeframes. Although trace clustering allows for the selective reconstruction of traces, it fails to mine the complete "history" (i.e. *evolution provenance*) of a business processes, identifying their diverse "tenancies" and how they differ one from the other.

2. **Process mining does not consider data flows.**
   Target meta-models for reconstruction typically consider the control flow of processes (i.e. the relation and sequence of process activities) and do not include data flows. As a result, reconstructed process models are not appropriate to serve as input for an information flow analysis. Security analysis is limited to control flow properties and cannot check data flow oriented properties.

2 Proposed Solution and Research Challenges

A first step on the way from process logs to meaningful security audits is the **development of time-based clustering algorithms** that are able to view the history of process structure changes during time. Reliable indicators for structural changes of processes have to be identified on the basis of process logs. The challenge here is that typically not every possible execution path of a workflow occurs with the same probability. In case of deviations from "typical" workflow behavior it is hard if not impossible to distinguish rare execution variants (traces) from traces related to a changed version of the workflow.

The essential part of this research plan is to **develop novel process mining techniques that incorporate data flow** aspects provided by process logs. This requires an analysis on which information can be extracted and which kind of meta-model is appropriate to describe data flows. In case of InDico [3] the meta-model is explicitly given and can be constructed by annotating reconstructed Petri

nets with data flow properties. Here the question is, if existing mining approaches can be extended to provide this functionality or if a complete redesign is required. On the other hand there may be other meta-models allowing different kinds of analysis. Which types of additional analysis techniques can be put on top of the proposed solution is another question picked up by this research plan.

Relying on process mining, the most relevant problem of this approach of finding a compromise between under- and overfitting a process log applies also for this research. A model is called to underfit a process log, if it allows too much behavior which is not reflected by the log. Algorithms showing this behavior have a tendency to over-generalize. Overfitting a log means to stay at a too specific level without any generalization. Such models typically establish sequential paths for each distinct trace and are merely another representation of the log.

2.1 Expected Impact

By introducing a novel trace clustering method, this research contributes to the fields *business provenance* that captures the lineage of business artifacts, including workflows [9]. Here, approaches to capture workflow evolution are missing. The proposed clustering technique *and* the subsequent mining should close this gap. It also contributes to *audit reduction* because the volume of audit records to facilitate manual review can be reduced. Auditors are able to select which among the various mined workflow specifications are to be tested for compliance with security policies.

Advancing process mining techniques to provide additional information about data flows forms a basis for applying manifold analysis techniques. Thus, this research plan also contributes to *security audits and CAAT*.

As an example the proposed techniques provide sufficient information for effective information flow analysis. Based upon InDico [2], the mined workflow specifications can be automatically tested for a multitude of security properties, including MAC-based, non-interference, and enterprise relevant properties, such as Chinese-wall and separation of duties. InDico aims to provide a well-founded, uniform approach and corresponding tool-support for the automated analysis of existing and/or mined workflow specifications for security properties. It defines a colored Petri net dialect, called IFnet, to model business process models and to formalize multi-level information flow properties [3].

Another possibility is to introduce a new form of analysis called contextualization. Combining the results of an information flow analysis showing covert channels, identified data flows can give evidence on their usage which enables much deeper checking of security requirements regarding information flow.

Allowing for a broader range of analysis techniques based on reconstructed process models, this research plan is an important contribution for the enhancement and automation of security audits in the BPM sector.

2.2 Preliminary Results

In order to find appropriate meta-models for efficient data- and information flow analysis, the starting point of this research was to analyze propagation graphs

capturing data flows within a process execution with extensional data flow properties, that denote what - instead of how - relevant industrial requirements are to be achieved [5]. Providing a sufficient basis for data flow analysis, propagation graphs do not reflect the control flow of process executions. While this first result was important to learn about the requirements of forensic analysis of business processes, it showed that alternative meta-models are needed to facilitate information flow analysis requiring both data flow and control flow.

A preliminary solution for the time-based clustering approach was already developed and presented at the *Workflow Security Audit and Certification* workshop at the BPM'11 conference [16]. It uses distances of process activities in the sense of intermediate activities to determine cluster borders. As long as the structure of a process does not change these distances move in fixed intervals. Changing operations cause boundary variations for at least one activity pair. Sequentially processing traces according to their timestamp, the algorithm uses samples to determine the typical behavior in terms of boundaries for the minimum and maximum observed value of activity-pair distances. Typical workflow behavior is determined on the basis of a parameter w (windowsize), that specifies the minimum number of traces used as "training" data and also defines the minimum cluster size. Using this clustering technique allows auditors to reconstruct the process history, so that they can appreciate the different ways in which a process evolves over time.

The appropriateness of *conformance checking* (process mining techniques providing proofs for the adherence of logged process traces to specific execution constraints) for security in the sense of CAATs was evaluated within a case study that will be presented at the Enterprise Engineering track of the ACM SAC conference in 2012.

3 Research Plan

The envisioned research is scheduled for a period of three years and divided into two packages picking up the aspects of section 2. Each package includes a phase where the applicability of the developed approaches is evaluated. Depending on the information about practical business process management and auditing that can be gained from expert interviews and cooperations with industry partners, evaluation will be either based on real process data (process models and logs) or on synthetic data having realistic characteristics.

1. Time-Oriented Clustering
   - Discovering suitable differentiation criteria within log files that support the detection of process changes
   - Development of new clustering algorithms that show process evolutions
   - Integration of developed algorithms in the security workflow analysis toolkit (SWAT) [4] developed at the University of Freiburg.
   - Conducting case-studies to evaluate approaches

2. Data Flow-Oriented Process Mining
   - Identification of data flow properties within log files
   - Reconstruction of IFnet-models
   - Integration of developed algorithms in SWAT.
   - Conducting case-studies to evaluate approaches

References

1. Accorsi, R.: Business process as a service: Chances for remote auditing. In: IEEE Computer Software and Applications Conference (2011)
2. Accorsi, R., Wonnemann, C.: Strong non-leak guarantees for workflow models. In: ACM Symposium on Applied Computing, pp. 308–314. ACM (2011)
3. Accorsi, R., Wonnemann, C.: InDico: Information Flow Analysis of Business Processes for Confidentiality Requirements. In: Cuellar, J., Lopez, J., Barthe, G., Pretschner, A. (eds.) STM 2010. LNCS, vol. 6710, pp. 194–209. Springer, Heidelberg (2011)
4. Accorsi, R., Wonnemann, C., Dochow, S.: SWAT: A security analysis toolkit for reliably process-aware information systems. In: Workshop on Security Aspects of Process-aware Information. IEEE
5. Accorsi, R., Wonnemann, C., Stocker, T.: Towards forensic data flow analysis of business process logs. In: Proceedings the IEEE Conference on Incident Management and Forensics. IEEE Computer Society (2011)
6. Adam, N., Atluri, V., Huang, W.: Modeling and analysis of workflows using petri nets. Intelligent Information Systems 10(2), 131–158 (1998)
7. Atluri, V., Warner, J.: Security for workflow systems. In: Handbook of Database Security, pp. 213–230 (2008)
8. Cummins, F.: BPM meets SOA. In: Handbook on Business Process Management 1. International Handbooks on Information Systems, pp. 461–479 (2010)
9. Curbera, F., Doganata, Y., Martens, A., Mukhi, N.K., Slominski, A.: Business Provenance – A Technology to Increase Traceability of End-to-End Operations. In: Meersman, R., Tari, Z. (eds.) OTM 2008, Part I. LNCS, vol. 5331, pp. 100–119. Springer, Heidelberg (2008)
10. de Medeiros, A.K.A., Guzzo, A., Greco, G., van der Aalst, W.M.P., Weijters, A.J.M.M., van Dongen, B.F., Saccà, D.: Process Mining Based on Clustering: A Quest for Precision. In: ter Hofstede, A.H.M., Benatallah, B., Paik, H.-Y. (eds.) BPM Workshops 2007. LNCS, vol. 4928, pp. 17–29. Springer, Heidelberg (2008)
11. Greco, G., Guzzo, A., Pontieri, L., Saccà, D.: Discovering expressive process models by clustering log traces. IEEE Transactions on Knowledge and Data Engineering 18(8), 1010–1027 (2006)
12. Lowis, L., Accorsi, R.: Finding vulnerabilities in SOA-based business processes. IEEE Transactions on Service Computing (2011) (to appear)
13. McHugh, J.: Handbook for the Computer Security Certification of Trusted Systems. Naval Research Laboratory (1995)
14. Sayana, A.: Using CAATs to support IS audit. Information Systems Control Journal, 1 (2003)
15. Song, M., Günther, C.W., van der Aalst, W.M.P.: Trace Clustering in Process Mining. In: Ardagna, D., Mecella, M., Yang, J. (eds.) BPM 2008 Workshops. LNBIP, vol. 17, pp. 109–120. Springer, Heidelberg (2009)

16. Stocker, T.: Time-Based Trace Clustering for Evolution-Aware Security Audits. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) BPM Workshops 2011, Part II. LNBIP, vol. 100, pp. 471–476. Springer, Heidelberg (2012)
17. Teeter, R., an Miklos Vasarhelyi, M.: Remote auditing: A research framework. Journal of Emerging Technology in Accounting (to appear)
18. van der Aalst, W., Weijters, T., Maruster, L.: Workflow mining: discovering process models from event logs. IEEE Transactions on Knowledge and Data Engineering 16(9), 1128–1142 (2004)
19. 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)
20. Wei, Y., Blake, M.: Service-oriented computing and cloud computing: Challenges and opportunities. IEEE Internet Computing 14, 72–75 (2010)

Adaptable UI for Web Service Composition: A Model-Driven Approach

Waldemar Ferreira Neto
Supervised by: Philippe Thiran

PReCISE Research Center, University of Namur, 5000, Belgium
{waldemar.neto,pthiran}@fundp.ac.be

Abstract. The main objective of this work is to provide User Interfaces (UI) for Web service compositions (WSC). We aim at investigating how user interfaces and their navigation can be derived from the WSC structures (data and control flows). We propose a model-driven engineering approach that provides models and transformational methods that allow deriving and adapting UI for any context of use.

Keywords: Web service composition, model-driven engineering, user interface, adaptation.

1 Introduction

Web services have gained attention due to the pressing need for integrating heterogeneous systems. A Web service is a software system designed to support interoperable machine-to-machine interactions over a network. It has an interface described in a machine-processable format. A main advantage of Web services is their ability of being composed. A Web service composition (WSC) consists in combining several Web services in a same process, in order to address complex user's needs that a single Web service could not satisfy [2].

There are several initiatives to provide languages that allow the description of a Web service composition. The current WSC languages are expressive enough to describe fully automated processes to build Web service compositions [2]. However, full-automated processes cannot represent all real-life scenarios specially those that need user interactions. In these scenarios, a user interaction may range from simple approvals to elaborate interactions where the user performs a complex data entry, for example, filling several forms.

Any computer system that involves users needs user interfaces (UI) to permit the interactions between the system and the user. The users of a WSC can interact with it through diverse devices (Desktop, Smart Phone, Tablet, among others) in diverse modalities (visual, aural, tactile, etc.). The adaptability of the UIs for a WSC has become necessary due to the variety of contexts of use.

In this work, we propose a model-driven engineering (MDE) approach for providing adaptable UIs from WSC. In particular, the approach relies on a modelization of user interactions within the WSC. Based on this modelization, the

G. Pallis et al. (Eds.): ICSOC 2011, LNCS 7221, pp. 177–182, 2012.

approach proposes a method to derive an abstract representation of the UI from a WSC. Interestingly, the derivation rules rely on the data/control flow of the WSC for specifying the navigation through the UIs. The obtained abstract representation can then be adapted to any specific context of use.

The remainder of this work is organized as follows. An overview of the works about user interactions and Web service composition is given in Section 2. Section 3 explores the research challenges associated with the generation of UI from WSC. Section 4 proposes an MDE approach to deal with challenges that were identified. Section 5 offers a preliminary plan for realizing our MDE approach and Section 6 concludes.

2 Related Work

There are several approaches that permit interactions between users and Web services. In some of these approaches, the information about the Web service (which can be WSDL or OWL-S) is used to infer a suitable user interface (e.g., [8]). To increase the usability of generated used interfaces, some approaches use additional information like UI annotation [9], platform-specific description [12], or user context [14]. In these approaches, the UI generation relies on type of the input and output described on the Web service description.

The development of Web interfaces for Web services has been addressed by the Web engineering community by the means of model-driven Web design approaches [15] and [4]. These approaches propose a model-based development method for process-based Web applications that use Web services. The former approach describes the Web service composition by BPMN and the UI navigation is described by a web-specific visual modeling language, WebML [4]. The latter relies on BPMN too, but the UI navigation is described on an object-oriented modeling language, OOWS [15]. Based on HTML templates, a set of UIs can be automatically generated form the WSC, and the navigation among these UIs is driven by the navigation model.

Another work that generates user interfaces for Web services is the Dynvoker [13]. This approach interprets a determined Web service and generates Web forms according to the Web service operation. Based on a BPEL-like language (GUI4CWS) this approach allows to handle complex service interaction scenarios. There are other approaches that allow a similar UI generation, but these approaches consider multiples actors [5] or/and context-aware UIs [11].

Other approaches generate UI for Web services based on the annotated Web service descriptions and the UI defined from a task model [17]. The annotations are used to generate the UI for the Web services and the task model drives the navigation among the UIs and Web services. As such, these approaches do separate the data/control flows of the WSC and the UI navigation model.

Other works aim at extending WSC descriptions with user interactions. An example of such extensions is BPEL4People [1], which introduces user actors into a Web service composition by defining a new type of BPEL activity to specify user tasks. However, this extension focuses only on the user task and

does not deal with the design of a user interface for the Web service composition. Another example of BPEL extensions that addresses the user interaction is BPEL4UI (Business Process Execution Language for User Interface) [6]. BPEL4UI extends the Partner Link part of BPEL in order to allow defining a binding between BPEL activities and an existing UI. This user interface is developed separately from the composition instead to be generated. In another work, Lee et al. [10] extend BPEL by adding interactive activities that are embedded in the BPEL code. Unlike BPEL4UI, this work specifies the UI together with the WSC, however the UI is specified for a unique context of use.

3 Research Challenges

The main objective of this work is to derive adaptable UI from WSC. In the following, we present the research challenges that must be tackled to achieve this objective.

First, we need to investigate how user interactions can be integrated within WSC. Concretely, WSC must be extended with user interaction activities that express the different possible types of user interactions [16]: data input interaction, data output interaction, data selection, and interaction by user event.

Another challenge is the fact that the navigation and the composition of the UI can rely on the control/data flow structures of the WSC extended with user interaction activities. A simple example of generation is given in Figure 1 that presents a simple travel reservation management. This WSC comprises three user interaction actives. The UI generation can lead to a UI grouping of the two first user interactions activities (*initializing Service* and *transportation means selection*) as data provided by these user interactions are mutually independent. However, this UI could not comprise the third user interaction activity (*providing license number*), as the user interaction will only be enable if the transportation means is the private car.

The last challenge is to be able to generate a UI adapted to the user context (user preference, user environment, and user platform) and the usability criteria (e.g., the size of the device screen).

4 Proposed Approach

We propose a Model-driven Engineering (MDE) approach that provides models and transformations for deriving and adapting UI from WSC and the context of use. We identify 3 main models and 3 main methodological steps.

4.1 Models

Our MDE approach relies on 3 models:

– *UI-WSC*: an extension of WSC with user interaction activities. To be compliant with current standards, the model is to rely on existing standards: a standard for WSC (e.g. BPEL) and a standard for describing user interfaces (e.g. UsiXML).

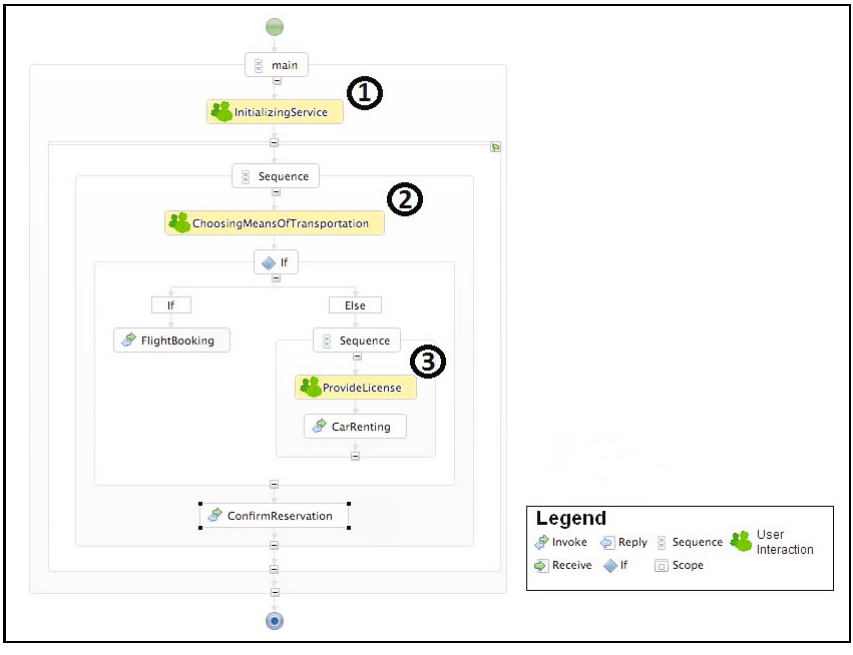


Fig. 1. Web service composition to manage travel reservations

- *Abstract user interface (AUI)*: this model describes the UI independently to any interaction modality (e.g. graphical modal, vocal modal) and computing platform (e.g. PC, smart phone). This model only specifies the UI components, their elements, and the navigation among the components.
- *Concrete user interface (CUI)*: this model is an adaptation of an AUI to a specific context of use (user preference, environment, and platform). For example, for visually handicapped person, an output abstract component could be transformed to a (concrete) vocal output.

4.2 Method

Our MDE method consists in 3 main steps:

- *Modeling*: where the WSCs are modelized within its user interactions by a designer using the UI-WSC.
- *Transformation*: where the AUI is derived by applying transformations to the UI-WSC model.
- *Adaptation*: where the CUI is derived from the AUI and the context of use. Additionally, the user can interact with the CUI through an interpreter, while a runtime component arbitrates the communication between the CUI and the WSC.

5 Research Methodology

The first part of our research consists in the definition of the different models of our MDE approach. In particular, we investigate and modelize how the user interaction can be specified within WSCs. Our goal here is to propose a extension to WSC meta-model (UI-WSC meta-model) with the user interaction activities representing the different possible types of user interactions. For the AUI and CUI meta-models, we refer to existing works in UI meta-modeling.

Next, we define the transformation rules for deriving an AUI description from a UI-WSC model. We plan to define these rules in an incremental way: starting with simple UI-WSC patterns (e.g., input/output sequence, choice) to continue with more complex ones (e.g., loop or interruptible area).

AUI adaptation is the next step. As there are existing approaches, we plan to investigate and evaluate these approaches so that we can to adopt the more suitable to our approach.

As a proof of concept, we develop a tool that not only supports the three main steps of your MDE method (design) but also orchestrate the WSC execution and the user interactions with the user (runtime).

Finally, we evaluate our approach. We first aim at evaluating our approach against other approaches (e. g. [6,15] and [4]). As comparison criteria, we adopt the usability criteria proposed by the ISO 9241 [7]: satisfaction, effectiveness, and efficiency. We also aim at evaluating our approach in real scenarios with real users.

6 Conclusion

In this work, we propose an MDE approach for providing adaptable UI from WSC. This approach aims at specifying all types of user interactions within WSC process, as well as the derivation of an abstract representation of the UI. The derivation rules rely on the data/control flow of the WSC for specifying the navigation through these abstract representations. Finally, the obtained representation can then be materialized to any specific context of use in order to provide an adapted UI.

So far, we have reviewed the literature about the users interactions and Web services. We have already proposed a BPEL extension able to modelize all types of user interactions within WSC processes, named UI-BPEL meta-model [3]. We have also implemented a design tool that is dedicated to edit a WSC conform to our UI-BPEL meta-model. The tool is an Eclipse plug-in based on the Eclipse BPEL Designer[1]. As future work, we plan to work on the transformation rules for deriving AUI from UI-BPEL and integrate these rules into our modeling tool.

References

1. Agrawal, A., Amend, M., Das, M., Ford, M., Keller, C., Kloppmann, M., König, D., Leymann, F., Müller, R., Pfau, G., et al.: Ws-bpel extension for people, bpel4people (2007)

[1] http://webapps.fundp.ac.be/wse/wiki/pmwiki.php?n=Projects.UIBPEL

2. ter Beek, M.H., Bucchiarone, A., Gnesi, S.: Web service composition approaches: From industrial standards to formal methods. In: ICIW, p. 15. IEEE Computer Society (2007)
3. Boukhebouze, M., Neto, W.P.F., Erbin, L.: Yet Another BPEL Extension for User Interactions. In: De Troyer, O., Bauzer Medeiros, C., Billen, R., Hallot, P., Simitsis, A., Van Mingroot, H. (eds.) ER Workshops 2011. LNCS, vol. 6999, pp. 24–33. Springer, Heidelberg (2011)
4. Brambilla, M., Dosmi, M., Fraternali, P.: Model-driven engineering of service orchestrations. In: Proceedings of the 7th Congress on Services, pp. 562–569. IEEE Computer Society, Washington, DC (2009)
5. Daniel, F., Casati, F., Benatallah, B., Shan, M.-C.: Hosted Universal Composition: Models, Languages and Infrastructure in mashArt. In: Laender, A.H.F., Castano, S., Dayal, U., Casati, F., de Oliveira, J.P.M. (eds.) ER 2009. LNCS, vol. 5829, pp. 428–443. Springer, Heidelberg (2009)
6. Daniel, F., Soi, S., Tranquillini, S., Casati, F., Heng, C., Yan, L.: From People to Services to UI: Distributed Orchestration of User Interfaces. In: Hull, R., Mendling, J., Tai, S. (eds.) BPM 2010. LNCS, vol. 6336, pp. 310–326. Springer, Heidelberg (2010)
7. ISO (ed.): ISO 9241-11: Ergonomic requirements for office work with visual display terminals (VDTs) – Part 9: Requirements for non-keyboard input devices (2000)
8. Kassoff, M., Kato, D., Mohsin, W.: Creating GUIs for web services. IEEE Internet Computing 7(5), 66–73 (2003)
9. Khushraj, D., Lassila, O.: Ontological Approach to Generating Personalized User Interfaces for Web Services. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 916–927. Springer, Heidelberg (2005)
10. Lee, J., Lin, Y.Y., Ma, S.P., Lee, S.J.: BPEL extensions to user-interactive service delivery. J. Inf. Sci. Eng. 25(5), 1427–1445 (2009)
11. Pietschmann, S., Voigt, M., Rümpel, A., Meißner, K.: CRUISe: Composition of Rich User Interface Services. In: Gaedke, M., Grossniklaus, M., Díaz, O. (eds.) ICWE 2009. LNCS, vol. 5648, pp. 473–476. Springer, Heidelberg (2009)
12. Song, K., Lee, K.H.: Generating multimodal user interfaces for web services. Interacting with Computers 20(4-5), 480–490 (2008)
13. Spillner, J., Feldmann, M., Braun, I., Springer, T., Schill, A.: Ad-Hoc Usage of Web Services with Dynvoker. In: Mähönen, P., Pohl, K., Priol, T. (eds.) ServiceWave 2008. LNCS, vol. 5377, pp. 208–219. Springer, Heidelberg (2008)
14. Steele, R., Khankan, K., Dillon, T.S.: Mobile web services discovery and invocation through auto-generation of abstract multimodal interface. In: ITCC (2), pp. 35–41. IEEE Computer Society (2005)
15. Torres, V., Pelechano, V.: Building Business Process Driven Web Applications. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) BPM 2006. LNCS, vol. 4102, pp. 322–337. Springer, Heidelberg (2006)
16. Trewin, S., Zimmermann, G., Vanderheiden, G.C.: Abstract representations as a basis for usable user interfaces. Interacting with Computers 16(3), 477–506 (2004)
17. Vermeulen, J., Vandriessche, Y., Clerckx, T., Luyten, K., Coninx, K.: Service-Interaction Descriptions: Augmenting Services with User Interface Models. In: Gulliksen, J., Harning, M.B., van der Veer, G.C., Wesson, J. (eds.) EIS 2007. LNCS, vol. 4940, pp. 447–464. Springer, Heidelberg (2008)

Economic Model Based Cloud Service Composition

Zhen Ye[1,2]
Supervised by: Athman Bouguettaya[3] and Xiaofang Zhou[1]

[1] The University of Queensland, Australia
[2] CSIRO ICT Centre, Australia
[3] Royal Melbourne Institute of Technology, Australia

Abstract. Cloud service composition is usually long-term based and driven by meeting the Cloud economic model. We use CP-Net to represent the long-term based economic model. We consider service composition as a Nash game to capture the behaviors of Cloud application providers and IaaS providers. Finally, we propose an economic model based service composition approach in Cloud computing.

1 Introduction

Cloud computing is becoming increasingly popular as the next-generation platform for conducting business. The main advantage of Cloud computing is its ability to scale to users' demands for storage, CPU and network resources [1]. The enabling technology for cloud computing is *Service Oriented Computing* (SOC)[2]. Cloud computing has *Cloud Providers* and *Cloud Consumers* like other traditional service-oriented platforms, such as Web service technology, Grid computing. Cloud service consumers are mostly *Cloud Application Providers*[1]. These consumers supply *Cloud Composite Applications* to end users. Cloud applications are mostly long-lived scalable applications subject to time-varying workload.

Compared to the traditional platforms, Cloud computing makes services more efficient and cost-effective for both providers and customers. Cloud service consumers, on one hand, save their cost by increasing and reducing resources at will. Cloud service providers, on the other hand, maximize the utilization of their resources and profits from the benefits of economics of scale. Indeed, Cloud computing introduces a new *Economic Model* for both consumers and providers.

We assume that the service infrastructure in Cloud computing is managed by a *Service Management System* [3]. The management system includes service description, service query and service composition etc. We focus on *Cloud Service Composition*. Service composition is the process of grouping/mashing up a set of services to provide a value added service. The result of this process is a composite service. This meets both the functional and non-functional (i.e., Quality of Service or QoS) requirements of consumers.

[1] In the remainder of the paper, we use *Cloud Consumer* and *Cloud Application Provider* interchangeably.

G. Pallis et al. (Eds.): ICSOC 2011, LNCS 7221, pp. 183–188, 2012.

Service composition in traditional platforms (e.g. SOC, Grid Computing) mostly assume it is short-term based and driven by meeting QoS at time of service query. In contrast, Cloud service composition is long-term based and driven by meeting the requirements of the economic model. In the Cloud economic model, computing resources, platforms and applications are virtualized as trading services in the market. Cloud services in the market are charged in a pay-as-you-go manner. Each Cloud application provider wants to maximise its profit while complying with QoS requirements, specified in Service Level Agreements (SLAs) with end users. Cloud service providers, in their turn, want to maximise the revenues obtained from providing the resources. We propose to use CP-Nets to represent the long-term based economic model for both Cloud application providers and Cloud service providers. Each Cloud application provider competes with others and bids for the use of Cloud services. The Cloud service provider competes with other Cloud service providers for obtaining more valuable consumers. In these competitions, the best choice of one depends on the choices of others. Therefore, we model both Cloud application providers and Cloud service providers as independent economic agents in a *Nash Game*. All players in the game attempt to selfishly optimise their utility. We then use the competition results to develop an efficient algorithm for Cloud service composition.

2 Composing Cloud Services

Cloud services refer to both the applications delivered as services over the Internet and the hardware and system software in the data centers that provide those services [2]. Cloud services are classified into three types: Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS).

For the sake of simplicity, we make several assumptions about Cloud services: 1) We only consider three types of IaaS during service composition. They are CPU services, network services and storage services. 2) Each SaaS provider is supported by a single PaaS provider. 3) Each PaaS provider is supported by a single IaaS provider. 4) Each IaaS provider supplies IaaS to multiple PaaS providers. 5) Each PaaS provider supplies PaaS to multiple SaaS providers. Cloud service composition, therefore, consists of compositions in three layers: SaaS composition, PaaS composition and IaaS composition. PaaS providers and IaaS providers are determined whenever a certain SaaS is selected for composing a composite service. We focus on SaaS composition and IaaS composition in this research.

2.1 Motivating Scenario

We use the scenario depicted in Fig. 1 to motivate and explain our approach. We consider an evaluation application. Universities, acting as Cloud application providers, supply evaluation applications to end users (e.g. tenure committee

members etc.). Based on the request workload on each university, the composition system aims to compose component SaaSs to form the evaluation composite applications. Each component SaaS further expects CPU, network and storage resources from IaaS providers.

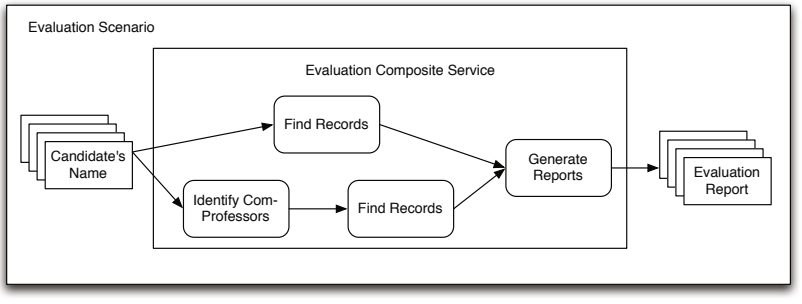


Fig. 1. Evaluation Cloud Service

2.2 Economic Model

The main part of the economic model is a representation of the long term based non-functional economic requirements. We use Conditional Preference Network (CP-Net) [4] to describe Cloud consumers' and Cloud providers' long term preferences over the non-functional properties.

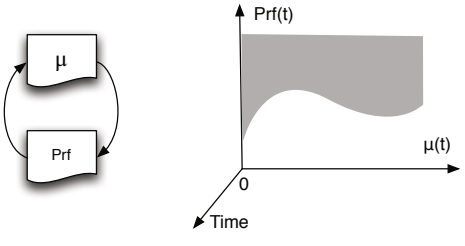


Fig. 2. Economic Model for a University

The economic model of a Cloud application provider depends on the SLAs established between the Cloud application provider and its end users. Each Cloud application provider usually serves different types of end users. The requests from different user types normally demand different composite applications both in functional and non-functional properties. Fig. 2 shows the economic model for a university in the example at any time t. The economic model includes two QoS attributes: $\mu_i(t)$ and $Prf_i(t)$. $\mu_i(t)$ represents the mean service rate of requests in university U_i at time t. $Prf_i(t)$ represents the total profit for university U_i from time t to $t + dt$. The universities always prefer larger $\mu(t)$ and $Prf(t)$. However, a larger $\mu(t)$ does not always lead to a larger $Prf(t)$. Hence, the university needs to clarify the satisfactory combinations of μ and $Prof$. The gray area in Fig. 2 represents the tolerant values for combination $[\mu(t), prf(t)]$ at time t.

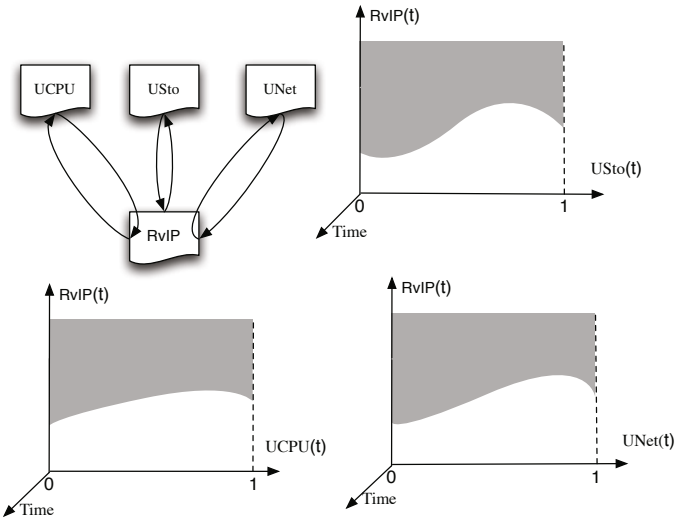


Fig. 3. Economic Model for an IaaS Provider

The economic model of an IaaS provider includes the non-functional requirements on the three types of Cloud services. SaaS obtain the three types resources, i.e., CPU, storage and network resources from the Cloud service provider in three ways [5]. Each of them defines a particular service market, as follows: 1) On-demand Market. This market allows resources to be supplied on demand at any time. The user is charged a usage fee, which is according to the time each resource is used. 2) Reservation Market. This market allows Cloud resources to be reserved for long periods of usage. It requires the payment of an upfront one time reservation fee. 3) Spot Market. Resources in this markets are offered in a best-effort way. Users access resources in this market by indicating the maximum usage fee that they are willing to pay. Fig. 3 shows the economic model for an IaaS provider in the example at any time t. The economic model includes four QoS attributes: utilization of CPU resources, storage resources and network resources, and the revenue. $UCPU_j(t)$, $USto_j(t)$ and $UNet_j(t)$ represents the utilization of CPU resources, storage resources and network resources in IaaS provider IP_j at time t. $RvIP_j(t)$ is the total revenue of IaaS provider IP_j from time t to $t + dt$. The IaaS provider aims to have higher utilization of its Cloud services, and higher revenue. Whereas, a higher utilization does not always lead to a higher revenue. The gray area in Fig. 3 represents the tolerant value combinations:$[UCPU(t), USto(t), UNet(t), RvIP(t)]$ at time t.

In the motivating scenario, each university would change its selection of SaaS and the unit number of IaaS according to its request workload. We assume that IaaS providers charge SaaS in terms of hourly basis [5]. Hence, each university faces the problem to satisfy its economic model by determining every hour the optimal component SaaS and the unit number of IaaS from each market. In addition, each IaaS provider's revenue and utilization of resources would vary

by changing the prices of its Cloud services. Each IaaS provider, therefore, faces the problem to satisfy its economic model by determining every hour the time unit price for reservation, on-demand resources and spot resources. The ultimate goal of the composition system is to satisfy the economic model for the Cloud system.

2.3 Cloud Service Composition Approach

In a Cloud system, the goal of a Cloud application provider is to determine every hour the number of reservation, on-demand and spot resources to execute its composite services to maximise its profit. Moreover, the goal of an IaaS provider is to determine the time unit cost for a unit number of on-demand, reservation and spot IaaS resources to maximise its total revenue. In this framework, Cloud application providers and IaaS providers are making decisions at the same time. The decisions of a Cloud application provider depend on those of other Cloud application providers and IaaS providers. Vice versa, the IaaS objective function depends on application providers' decisions and other IaaS providers.

We model Cloud service composition as a *Nash Game* [6] where Cloud application providers and Cloud service providers are players in the game. In the Nash game, a set of strategies for the players constitute a *Nash Equilibrium* if no player can benefit by changing its strategy while other players keep their strategies unchanged. Following this Nash equilibrium concept [7], Cloud application providers and IaaS providers adopt a strategy such that none of them can improve its revenue by changing its strategy unilaterally. At the end of this competition (game), there might exist more than one equilibrium. The equilibrium which leads to a Cloud system that satisfies the economic model are the final solution for the Cloud service composition problem.

3 Related Work and Summary

Most composition approaches in traditional platforms (e.g. SOC) use linear programming methods. Two composition approaches are proposed in [8]. One of them focuses on local optimization. The other approach focuses on global optimization. They use integer programming to solve the optimization problem. Ardagna et al. [9] further propose an improved approach. This approach uses Mixed Linear Programming (MILP). They also introduce several concepts such as loop peeling and negotiation mechanisms to address situation where no feasible solution can be found. Alrifai and Risse [10] propose an approach to decompose global QoS constraints into local constraints with conservative upper and lower bounds. These local constraints are resolved by using an efficient distributed local selection strategy. Recently, several approaches have been proposed to solve service-related problems in Cloud Computing. Wu et al. [11] propose a dynamic workflow-based resource allocation approach to enhance the overall performance of composite services in Cloud computing. Ardagna et al. [6] model the service provisioning problem as a generalized Nash game, and proposes an efficient algorithm for the run time management and allocation of IaaS

resources to competing SaaSs. However, the problem of service composition has not been considered in Cloud computing platform.

We adapt an economic model based approach to solve Cloud service composition problems. Compared to traditional service composition approaches, our work leverages two difficulties when considering service composition in Cloud computing. Cloud service composition is driven by meeting the Cloud economic model. In addition, Cloud service composition is usually long-term based. Hence, we adopt CP-Net to represent the long-term based economic model. Service composition needs to be conducted every hour to meet the changes. Furthermore, We consider service composition as a Nash game to capture the behaviors of Cloud application providers and IaaS providers. We finally use the results to develop an efficient algorithm for Cloud service composition.

References

1. Motahari-Nezhad, H., Stephenson, B., Singhal, S.: Outsourcing Business to Cloud Computing Services: Opportunities and Challenges. LABs of HP (2009)
2. Armbrust, M., Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., et al.: Above the clouds: A berkeley view of cloud computing. EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28 (2009)
3. Yu, Q., Liu, X., Bouguettaya, A., Medjahed, B.: Deploying and managing web services: issues, solutions, and directions. The VLDB Journal The International Journal on Very Large Data Bases 17(3), 537–572 (2008)
4. Boutilier, C., Brafman, R., Domshlak, C., Hoos, H., Poole, D.: Cp-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. Journal of Artificial Intelligence Research 21(1), 135–191 (2004)
5. Amazon elastic compute cloud pricing (2011), http://aws.amazon.com/ec2/pricing/
6. Ardagna, D., Panicucci, B., Passacantando, M.: A game theoretic formulation of the service provisioning problem in cloud systems. In: Proceedings of the 20th International Conference on World Wide Web, pp. 177–186. ACM (2011)
7. Facchinei, F., Kanzow, C.: Generalized nash equilibrium problems. 4OR: A Quarterly Journal of Operations Research 5(3), 173–210 (2007)
8. Zeng, L., Benatallah, B., Ngu, A., Dumas, M., Kalagnanam, J., Chang, H.: QoS-aware middleware for web services composition. IEEE Transactions on Software Engineering 30(5), 311–327 (2004)
9. Ardagna, D., Pernici, B.: Adaptive service composition in flexible processes. IEEE Transactions on Software Engineering, 369–384 (2007)
10. Alrifai, M., Risse, T.: Combining global optimization with local selection for efficient qos-aware service composition. In: Proceedings of the 18th International Conference on World Wide Web, pp. 881–890. ACM (2009)
11. Wu, B., Chi, C., Chen, Z., Gu, M., Sun, J.: Workflow-based resource allocation to optimize overall performance of composite services. Future Generation Computer Systems 25(3), 199–212 (2009)

Towards Opportunistic Service Composition in Dynamic Ad Hoc Environments

Christin Groba*
Supervised by: Siobhán Clarke

Lero Graduate School of Software Engineering
Distributed Systems Group
School of Computer Science and Statistics
Trinity College Dublin, Ireland
{grobac,Siobhan.Clarke}@scss.tcd.ie

Abstract. Service composition in pervasive computing envisions the collaboration of services provided by mobile and embedded devices to achieve complex application requirements. Although some research has investigated the composition of services in dynamic ad hoc settings, it is not clear how service composites manage the conflict between application layer complexities and network unreliability that may cause them to fail repeatedly. This paper outlines investigations on opportunistic service composition as a solution for managing complex service requests in mobile infrastructure-less environments. Opportunistic service composition dynamically transfers control among service providers to bind and execute services in a decentralised way. Unlike existing fragmentation and broker-based designs, it interleaves provider allocation with service execution to counteract mobility in a flexible manner. Early simulation results suggest that this approach reduces the failure probability and communication overhead of complex service requests. This illustrates that opportunistic service composition may be feasible to manage collaboration in dynamic ad hoc environments.

1 Introduction

Advances in mobile and embedded device technology have led to the vision of pervasive computing environments in which complex applications can be achieved by composing the services and capabilities of different devices [2]. Service composition is the process of discovering, selecting, and invoking suitable service providers based on a description that defines the order and nature of required services. A service composite represents such an arrangement of constituent services. The notion of combining services that are offered by different devices is only just emerging in environments that lack a managed infrastructure and a stable network topology [6]. The following scenario motivates an example for service

* Partially supported by Science Foundation Ireland grant 03/CE2/I303_1 to Lero - the Irish Software Engineering Research Centre (http://www.lero.ie).

G. Pallis et al. (Eds.): ICSOC 2011, LNCS 7221, pp. 189–194, 2012.

composition in ad hoc settings, where dynamics mainly stem from node mobility. Microphone sensors on mobile phones may soon allow for personal health monitoring such as tracking daily exposure to noise by periodically recording and analysing audio snippets. Receiving or making a call, however, blocks noise recording and results in a non-equidistant data series which could hamper further analysis. Leveraging nearby mobile phones that temporarily share their audio capabilities may be one solution to this problem. If aggregated context information rather than raw data values are required, audio tracking is complex and entails multiple sub-tasks (e.g., sampling, filtering, classifying, and geo-tagging). A single surrogate device may not have the resources or suitable software to complete the tasks alone and thus requires the collective effort of multiple peers. However, this collaboration may fail if peers are unable to communicate due to a) their mobility and thus dynamic availability or b) repeated path or message loss in the network.

2 Problem Statement

Coordinating multiple parties to accomplish service composites in dynamic ad hoc environments is challenging due to the conflict between application complexities and network limitations. Ad hoc or hastily formed networks lack a central entity that has a global system view and the ability to undertake service discovery and execution alone. Service composition has to be decentralised due to the nature of the environment which in turn requires additional coordination and synchronisation among the participators. For composites with forking and merging tasks, for example, multiple service providers have to synchronise and agree on a common merge provider. In addition, system knowledge about available services has first to be established. Routes between service providers initially do not exist because service providers are not aware of their neighbours until they announce themselves or respond to a request. Generally, decentralisation, the need for synchronisation, and acquisition of system knowledge requires service providers to communicate. Communication in ad hoc networks, however, is wireless and inherently unreliable causing message loss and failure. Mobile service providers increase the probability of failure because they frequently change the network topology creating disconnections and stale routes. Mechanisms for repair and recovery produce extra network traffic and strain the limited wireless bandwidth even more.

For these reasons, this work investigates the ***research question*** of what strategies allow for service composition in dynamic ad hoc environments and at the same time keep communication over a resource-constrained, unreliable network low. ***Key challenges*** of this research are: First, communication has to be reduced because the more the composite is exposed to an unreliable network, the more likely it is to fail. Second, service discovery in initially unknown settings nonetheless requires the exchange of information. Third, service flows, i.e., composites with splitting and merging tasks, need synchronisation among potentially unknown parties, which again requires communication.

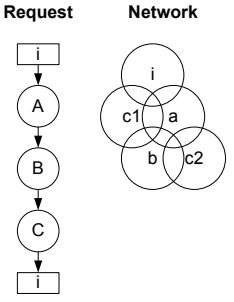
3 Proposed Solution

This work proposes an opportunistic composition protocol to satisfy complex service requests in mobile ad hoc networks. It is designed to reduce the failure probability and communication overhead of service composites. A service provider temporarily acquires the role of the coordinator who executes its local service and finds a provider for the next required service. Control over provider allocation and service execution travels from one provider to the next achieving full decentralisation. The protocol interleaves the allocation process and executes a provider immediately after it has been bound. Binding a provider only prior to its execution gives it less time to move disadvantageously and thus decreases the probability of stale routes and the need for recovery.

Late binding implies that providers are searched and selected at runtime. The proposed solution employs a directory-less architecture in which nodes are only aware of their own services. Service providers advertise themselves if they overhear a composite request and know they offer the next required service. If they provide a service at a later stage, they observe the composition and hold their advertisement back until their service is required next. This situational service advertisement avoids constant network traffic created by otherwise periodically beaconing service providers. Similarly, it does not risk traffic peaks as beaconing would when many providers simultaneously arrive in a new location. For selection, the providers' connectivity status serves as a decision criterion, if multiple providers offer the same service.

Complex application requirements provide an additional opportunity to reduce network traffic because handling them creates interaction among network nodes and thus system knowledge. Little or no extra routing messages are needed to keep an up to date view of the network topology. Routing information can be deduced from the underlying composition protocol and by observing the message exchange among service providers.

These strategies and architecture decisions address the first two research challenges. First, decentralised interleaved service allocation and execution reduces the composite's exposure to the unreliable network. Second, situational service advertisement and observation of the composition process allow for efficient service discovery in initially unknown settings. The example in Figure 1 illustrates these strategies. A client or composition initiator i issues a request with three sequentially-connected services A, B, C, and specifies that it wants to receive the overall composition result. The network consists of nodes whose lower-case node identifiers indicate that they host the corresponding upper-case service (e.g., node a hosts service A). The protocol consists of three message types: *Search* contains the remaining part of the request that has not been executed yet. *Ad* advertises that a node provides a required service and the way it is connected. *Token* transfers the coordinator role to the contained service and its provider. In addition to the protocol comments, note that a node knows about its connections only if it received a message from a neighbour. Therefore, when node a advertises itself, it is not aware of other neighbours except from the initiator i. The example shows how the request travels from one node to the next and how



Protocol

Sender	Message	Content	Comment
i	search	A-B-C-i	// i issues complex service request
a	ad	A/a, con(i)	// a provides A and connects i
i	token	A/a	// i binds a and transfers control to a
			// **a executes A locally**
a	search	B-C-i	// no provider for B advertised itself
b	ad	B/b, con(a)	// b provides B and connects to a
a	token	B/b	// a binds b and transfers control to b
			// **b executes B locally**
c1	ad	C/c1, con(i,a,b)	// c1 advertises due to search from i, a
c2	ad	C/c2, con(a,b)	// c2 advertises due to search from a
b	token	C/c1	// b binds c1 due to better connectivity
			// **c1 executes C locally**
c1	token	i	// c1 returns composite result to i

Fig. 1. Example protocol for composing sequentially-connected services

service execution interleaves the allocation process. The behaviour of node c1 illustrates situational service advertising as it does not instantly respond to i's search. Binding service C demonstrates connectivity-based provider selection.

The third research challenge addresses service flows. Following the late binding approach means that the merge provider is bound only prior to its execution. This, however, is challenging because parallel tasks imply that the coordinator role is assigned to multiple providers at the same time. These concurrent controllers may be mutually unknown and nonetheless have to identify and agree on the same merge provider. Current work explores how tracking the composition progress and connectivity status can help to find a common merge provider efficiently. In addition to that, this work investigates how service flows can be dynamically adapted to match the current provider availability.

Earlier work [3] studied how interleaving service allocation and invocation compares to an approach that first allocates all service providers and only starts executing after the last required service has been bound. ***Simulation results*** in Figure 2 show for sequentially-connected sub-services that the proposed protocol generally improves the composition success ratio, response time, and communication effort in comparison to the baseline approach.

4 Progress beyond the State-of-the-Art

Key approaches for achieving complex service requests can be classified into centralised, hierarchical, and decentralised composition management.

Centralised solutions employ a single coordinator that may be fixed or dynamically elected to allocate and invoke required services. A fixed coordinator [4] can cause long expensive routes, if it is not in vicinity of the mobile service providers. A dynamically appointed coordinator or broker [2] is more flexible because it is assigned afresh for each new request and selected based on how well it connects to required services. However, it still requires all messages to be passed through it preventing direct interaction among providers.

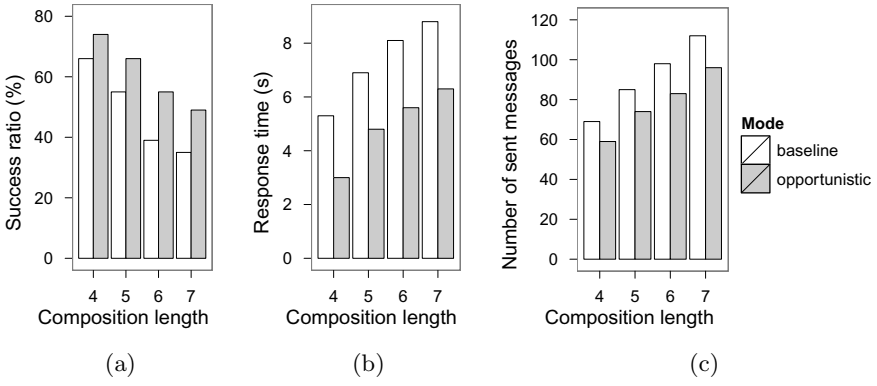


Fig. 2. Simulation results for sequentially-connected services moving at 1-13 m/s

Hierarchical architectures [1] relax this requirement and enable direct message exchange among service providers. The coordinator solely allocates services and monitors their execution. However, the coordinator requires all service providers to stay connected and to achieve this, it controls their movement. Unstructured settings in which service providers act independently may not allow for such an authority.

Decentralised approaches may contain a central component but its responsibility is limited to service discovery [5] or provider selection [6] leaving service execution and movement decisions entirely to the service providers. Other composition solutions, like [8] or the one proposed in this paper, discover, select, and execute service providers fully decentralised. Provider allocation further distinguishes all these decentralised approaches. Fragmentation-based designs [6] bind all services prior to execution. This raises two issues: First, they bind providers that may not get executed due to conditional tasks or premature termination allocating scarce resources in vain. Second, composites are less sensitive to providers that dynamically arrive and depart during execution. Designs supporting decentralised provider selection are more flexible. They bind providers during execution and only treat service flows in a special way. Merge tasks are bound either before composite execution starts [5] or early during execution together with the fork task [8]. However, the merge provider is assumed to be reachable from all parallel branches and to be available until it gets executed. This assumption may not always hold in mobile ad hoc networks. A solution for binding the merge node only right before it should execute [7] incurs a high communication overhead that seems infeasible for bandwidth-constrained networks: For each service in each branch the binding decision needs to be exchanged with all other branches. In addition, each branch selects its next provider independent from all other branches and thus risks long routes between the final decision partners.

Fully decentralised service composition with support for late binding for all sub-services has not yet been investigated in dynamic, ad hoc, and bandwidth-constraint networks. The proposed work targets this gap and studies what effect opportunistic service composition has in such settings. Its aim is to contribute to a better understanding of what service composition approaches are available and how they may be applied in mobile infrastructure-less environments.

5 Research Plan

Current work explores ways of extending the proposed protocol to handle service flows. Enabling service providers to observe the composition looks promising for binding a merge provider late and efficiently. Suitable recovery strategies will be required for compositions that started but cannot finish because a provider for a required service does not exist. The verification of protocol properties requires a formal notation of the protocol. UML state charts and Markov chains are at present analysed to assess their suitability for modelling the dynamic controller role and the gradually advancing topology knowledge. Further evidence on the feasibility of the opportunistic approach in dynamic ad hoc environments will be collected by conducting experiments. The simulation of mobile infrastructure-less networks creates a controlled environment that allows for comparing runtime characteristics of the proposed approach against related solutions.

References

1. Catarci, T., de Leoni, M., Marrella, A., Mecella, M., Salvatore, B., Vetere, G., Dustdar, S., Juszczyk, L., Manzoor, A., Truong, H.L.: Pervasive software environments for supporting disaster responses. IEEE Internet Computing 12(1), 26–37 (2008)
2. Chakraborty, D., Joshi, A., Finin, T., Yesha, Y.: Service composition for mobile environments. Mobile Networks and Applications 10, 435–451 (2005)
3. Groba, C., Clarke, S.: Opportunistic composition of sequentially-connected services in mobile computing environments. In: International Conference on Web Services (ICWS), pp. 17–24. IEEE (2011)
4. Philips, E., Van Der Straeten, R., Jonckers, V.: NOW: A Workflow Language for Orchestration in Nomadic Networks. In: Clarke, D., Agha, G. (eds.) COORDINATION 2010. LNCS, vol. 6116, pp. 31–45. Springer, Heidelberg (2010)
5. Schuler, C., Weber, R., Schuldt, H., Schek, H.J.: Scalable peer-to-peer process management - the osiris approach. In: International Conference on Web Services (ICWS), pp. 26–34. IEEE (2004)
6. Sen, R., Roman, G.-C., Gill, C.D.: CiAN: A Workflow Engine for MANETs. In: Wang, A.H., Zavattaro, G. (eds.) COORDINATION 2008. LNCS, vol. 5052, pp. 280–295. Springer, Heidelberg (2008)
7. Yildiz, U., Godart, C.: Synchronization solutions for decentralized service orchestrations. In: International Conference on Internet and Web Applications and Services (ICIW), p. 39. IEEE (2007)
8. Yu, W.: Decentralized Orchestration of BPEL Processes with Execution Consistency. In: Li, Q., Feng, L., Pei, J., Wang, S.X., Zhou, X., Zhu, Q.-M. (eds.) APWeb/WAIM 2009. LNCS, vol. 5446, pp. 665–670. Springer, Heidelberg (2009)

Model Checking Inconsistency Recovery Costs

Gang Zhou
Supervised by: Heinz Schmidt, Ian Peake, and Lawrence Cavedon

School of CSIT, RMIT University, Melbourne, Australia
g.zhou@student.rmit.edu.au

Abstract. Typically, when services become inconsistent from a business viewpoint, it is expected that compensation be used to recover from the inconsistency, by undoing the executed operations. In reality, compensation may incur additional costs, however existing approaches to recovery do not take such costs fully into account. We identify some major underlying gaps in SOC (Service Oriented Computing) related to compensation modelling, inconsistency identification, recovery and cost calculation. To make services more reasonable and predictable in dealing with inconsistencies from a cost perspective. We propose a cost-aware compensation framework modelling service compensations and compositions by a Petri Net-based model, reasoning about inconsistency recovery behaviours by model checking the *LTL* properties corresponding to business rules, and computing costs of recovery behaviours by parameterised cost calculation.

1 Introduction

Services are distributed self-describing open units that can be recursively composed for meeting business goals[25]. During transactions, services may become inconsistent with respect to business rules due to exceptions such as service failure, network error and human behaviours. Database transaction management[18] for resolving data integrity inconsistency is not suitable in SOC as the 2PC (2 Phase Commit) cannot be realised[19,15]. Handling compensation is firstly studied in the context of *sagas*[17], in which a compensable transaction consists of a sequence of smaller ACID transactions[17,19]. Nowadays, compensation is extensively applied in SOC to maintain consistencies by undoing the executed operations in case of transaction failures [26,8].

In reality, however, compensation may incur costs such as time, efforts and money etc. It is challenging for a service provider to determine such costs, as exceptions at any time may lead to different service inconsistencies that are the service's states and behaviours where business rules are violated. Recovery from different inconsistencies may incur different costs. For example, a customer may cancel an order from an online shop after the goods has been packed, where the inconsistency is identified as $goods\_packed \wedge order\_canceled$. To recover this inconsistency, the action *unpack_goods* is invoked to undo the effects of the *pack_goods* and costs y man-hours. Furthermore, the costs vary when the cancelation is raised at different time due to different compensation behaviours involved

G. Pallis et al. (Eds.): ICSOC 2011, LNCS 7221, pp. 195–200, 2012.

e.g. the order is canceled after the goods have been shipped, then the compensation may include *reshipping* and *unloading* which may cost more resources.

In concurrency, the cost of compensation-based recoveries depends on not only the compensation behaviours, but also the types of resource being consumed. For instance, the time costs of two concurrent actions may be computed by *max*, while the financial costs are summed; however for *choice* actions, the overall time and money costs are both computed by *max*. Hence cost calculation must be parameterised with respect to different kinds of composition and resources.

Moreover, cost-effective recoveries should be automated. Thus there is a clear need for a formal architecture framework to reason about the costs of compensation-based service inconsistency recovery. In such a framework, services inconsistencies would be handled more reasonably and predictably, e.g. by determining the inconsistency recovery behaviours and the minimal cost of such behaviours. At present, no such framework exists, due to some unsolved fundamental gaps such as the lack of a formal model of service composition bundled with compensation with supporting inconsistency identification and recovery, and the lack of a method for cost calculation parameterised with respect to different kinds of compositions and resources.

In this research, we will propose a formal framework to model and reason about costs of compensation-based service inconsistency recoveries. Motivated by the gaps above, we propose a research program as follows. Firstly, we will formalise an extended Petri net model modelling service composition, compensation and costs. Secondly, we will describe business rules in terms of *LTL* [29] properties and use model checking to identify inconsistency recoveries. Finally, we will reflect these recoveries into the Petri net model to calculate costs.

The remainder of this paper is organised as follows. Current issues are identified by literature review in section 2. The overview of the approach is sketched in section 3. Research plan and ongoing work are briefed in section 4 presents. Finally we conclude with a discussion in section 5.

2 Related Work and Research Gaps

Current models and methods cannot fully handle costs in compensation-based service inconsistency recoveries due to three major fundamental gaps. Firstly, ***compensation in SOC is not formally modelled***. Compensations in SOC are not formalised and automated in existing web service frameworks [5,13,10,27] in which developers need to program the *compensation handlers* to deal with ad hoc inconsistencies. Existing formal transaction models such as *sagas* [17], *StAC* [11], *t-calculus* [22], *cCSP* [12] and *cWFN* (Compensational Workflow Net) [2] that formally describe compensations in LRTs (Long Running Transactions) are not applicable directly in SOC, because services' interfaces and message-exchange based compositions are not considered. On the other hand, some models formalise services but do not take compensations into account, such as *oWFN* (open Workflow Net) [28] suggested by Reisig and service automata [24]. Secondly ***there is no support for reasoning about inconsistency recovery***

behaviours. Existing frameworks for ensuring service states based consistencies such as *WTDP* [14], *set consistency* [16] and *webservice interfaces* [6] are concerned only with the consistencies of end states. However, it is overlooked that the inconsistencies may be tolerated and recovered eventually before a service ends. Finally, ***parameterised cost calculations are not studied***. Costs in compensations discussed by Biswas [8] and Li [23] are not formalised. Timed automata [3] and weighted timed automata [4] formalise the linear costs based on state transition models in which actions are interleaved and studied in sequences. Consequently, *truly concurrency* and resource types introduced above are not included. Some timed Petri net based models [20,1] reason about linear time costs, however, they are based on reachability graphs, that are converted interleaved models where information about concurrency has been lost.

3 Overview of the Proposed Approach

We propose a novel approach to tackle the overall gaps. For modelling compensation in SOC environments, a Petri Net based model is proposed due to its algebraic and graphic capabilities, and the elegance of describing *true concurrency*. We propose the CSN (Compensable Service Net) by combining *cWFN* and *oWFN*. The former models normal tasks, failure tasks and compensation tasks; the latter models services interfaces and compositions. Thus we combine their strengths including cost modelling. For model checking inconsistency, a business rule denoted by φ is defined by two *LTL* properties separately: the *safety* property $G\varphi$ to ensure φ is satisfied in all states (markings) and the end-state property $F\varphi$ to ensure φ is met in the end. We define an inconsistency of a service as a service behaviour that violates the *safety* property of a business rule. By model checking these properties on every execution trace, inconsistency recovery paths are reasoned from counterexamples if any. For cost calculation, first inconsistent states and final consistent states obtained from the recovery paths are reflected to CSN and costs are calculated in the net.

Service Compensation Modelling. In CSN, all actions including communication, normal, failure or compensation actions are represented by transitions while the preconditions (effects) of actions are represented by places. Four special places such as *receiveIn*, *successOut*, *receiveCancel* and *sendAbort*, called *interfaces places* are distinguished for modelling communications with environments including e.g. other CSNs. A number of algebraic operators defined in [22], e.g. *sequence* and *parallel* etc. are translated to the service composition operators for CSNs' compositions by gluing all shared *interface places*, therefore, both normal and the compensation processes are constructed.

Model Checking Inconsistency. We propose a model checking procedure which has the inputs of a finite set of execution paths extracted from the reachability graph of CSN and the two *LTL* properties: *safety* and *end-state*, from business rules; its output is a set of recovery paths in each of which a first inconsistent state and a final consistent state will be reflected back to the CSN

for costs calculations. This procedure is briefed as follows. For each execution trace, model checking both *LTL* properties, if the end state consistency is satisfied and the *safety* consistency is violated, then we locate the last element in the counterexample, which is the first action breaking the *invariance*, the consistency specification. Therefore, the recovery path in this execution is identified from the located action to the final action. This procedure is recursively applied on the sub CSNs to find out all sub paths that record all inconsistencies' tolerances. For the sub CSN model checking, the properties are refined in terms of the sub CSN's observable actions, thus the specification decomposition must be considered.

Parameterised Cost Calculation. Resources and their quantities are represented by a set of special places called *resource places* and the tokens within the places respectively. A cost function of a transition is derived from its firing rule by projecting only the arcs connecting with *resource places*. Therefore, given a parameterised cost specification that represents the cost calculation rule for a resource type, for instance, time cost is computed by *max* for parallel transitions, and by + for sequence transitions, the costs from one state to another in a CSN system can be decidable with some restrictions and the algorithm is currently under development.

4 The Research Plan and Current Work

This ongoing research is planned in three stages according to the proposed approach and correspondingly, the framework consists of three major modules: Service Net Compositor (SNC), Inconsistency Recovery Model Checker (IRMC) and Parameterised Cost Calculator (PCC) (see Fig. 1). In the first stage, we define and implement the CSN with a number of composition operators. Hence, the formal model of service composition bundled with compensation is defined and implemented in SNC. In the second stage, the IRMC module is realised in which all compensation-based recovery paths from inconsistencies are identified by model checking the *LTL* inconsistency properties on a reachability structure that is derived from a CSN. In the last stage, parameterised costs are calculated in the PCC.

The plan is carried out incrementally and iteratively and we have implemented the first two stages in a prototype applying several case studies. The SNC is built on *PIPE* [9]. The sub CSNs are input in the forms of *PNML* [7] files and their operators are input as the algebra operators. These operators i.e. *parallel*, *sequence* and *choice* have been translated and implemented to the net composition operators. More operators will be considered when necessary e.g. *alternative forwarding* and *iteration*. The *LTL* properties are generated according to business rules input by users and checked in the IRMC module which has been built based on *LTSA* [21]. All inconsistency recovery behaviours are output by IRMC and will be provided to PCC for computing costs.

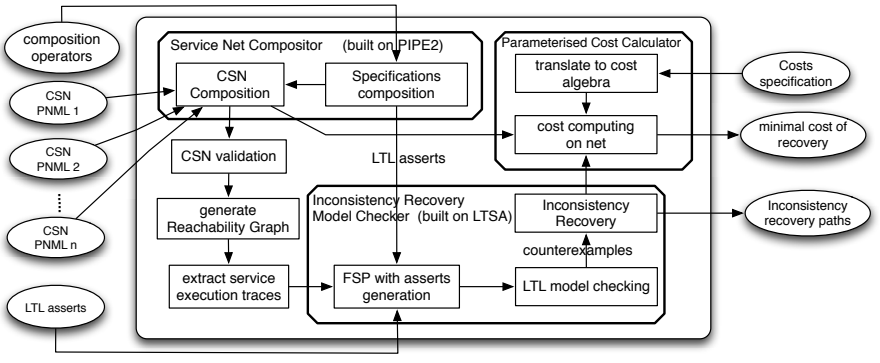


Fig. 1. Compensation-based inconsistency recovery and costs reasoning framework

5 Discussion

In SOC, compensation with costs to recover from inconsistencies is not formally studied. We identify three fundamental gaps related to compensation modelling, inconsistency recovery and parameterised cost calculation. The contribution of this paper is to propose a formal framework tackling all these gaps, reasoning about costs of compensation-based solutions. The inconsistencies captured by this research are the services' states and behaviours that violate the *safety* property of business rules. In some cases, not all inconsistencies can be recovered even when they are detected and these situations should be diagnosed. We are scoping the inconsistencies that can be recovered. Complex case studies suggest we must face and handle the well-known state space explosion problem. The efficiency of model checking algorithms can be improved by reducing the redundant computations on identifying the same inconsistencies. The cost model needs to be enhanced to support resource sharing and racing problems in the future.

References

1. Abdulla, P.A., Mayr, R.: Minimal Cost Reachability/Coverability in Priced Timed Petri Nets. In: de Alfaro, L. (ed.) FOSSACS 2009. LNCS, vol. 5504, pp. 348–363. Springer, Heidelberg (2009)
2. Acu, B., Reisig, W.: Compensation in Workflow Nets. In: Donatelli, S., Thiagarajan, P.S. (eds.) ICATPN 2006. LNCS, vol. 4024, pp. 65–83. Springer, Heidelberg (2006)
3. Alur, R., et al.: A theory of timed automata. Theoretical Computer Science 126(2), 183–235 (1994)
4. Alur, R., La Torre, S., Pappas, G.J.: Optimal Paths in Weighted Timed Automata. In: Di Benedetto, M.D., Sangiovanni-Vincentelli, A.L. (eds.) HSCC 2001. LNCS, vol. 2034, pp. 49–62. Springer, Heidelberg (2001)
5. Andrews, T., et al.: Business process execution language for web services, version 1.1. Standards proposal by BEA Systems, International Business Machines Corporation, and Microsoft Corporation (2003)
6. Beyer, D., Chakrabarti, A., Henzinger, T.A.: Web service interfaces, pp. 148–159 (2005)

7. Billington, J., Christensen, S., van Hee, K.M., Kindler, E., Kummer, O., Petrucci, L., Post, R., Stehno, C., Weber, M.: The Petri Net Markup Language: Concepts, Technology, and Tools. In: van der Aalst, W.M.P., Best, E. (eds.) ICATPN 2003. LNCS, vol. 2679, pp. 483–505. Springer, Heidelberg (2003)
8. Biswas, D.: Compensation in the World of Web Services Composition. In: Cardoso, J., Sheth, A.P. (eds.) SWSWPC 2004. LNCS, vol. 3387, pp. 69–80. Springer, Heidelberg (2005)
9. Bonet, P., et al.: PIPE v2. 5: A Petri net tool for performance modelling. In: Proc. 23rd Latin American Conference on Informatics (CLEI 2007). Citeseer (2007)
10. Bunting, D., et al.: Web Services Composite Application Framework (WS-CAF). Ver1. 0 (2003)
11. Butler, M., Ferreira, C.: A process compensation language, pp. 61–76 (2000)
12. Butler, M., Hoare, T., Ferreira, C.: A Trace Semantics for Long-Running Transactions. In: Abdallah, A.E., Jones, C.B., Sanders, J.W. (eds.) CSP25. LNCS, vol. 3525, pp. 133–150. Springer, Heidelberg (2005)
13. Cabrera, F., Copeland, G., Cox, B., Freund, T., Klein, J., Storey, T., Thatte, S.: Web services transaction (WS-transaction). In: Microsoft, IBM etc (2002)
14. Choi, S., et al.: A framework for ensuring consistency of Web Services Transactions. Information and Software Technology 50(7-8), 684–696 (2008)
15. Coleman, J.: Examining BPEL's compensation construct. In: Rigorous Engineering of Fault-Tolerant Systems (REFT 2005), p. 122 (2005)
16. Fischer, J., Majumdar, R.: Ensuring consistency in long running transactions. In: Proceedings of the Twenty-Second IEEE/ACM International Conference on Automated Software Engineering, pp. 54–63. ACM (2007)
17. Garcia-Molina, H., Salem, K.: Sagas. ACM SIGMOD Record 16(3), 249–259 (1987)
18. Gray, J.: The transaction concept: Virtues and limitations. In: Proceedings of the Very Large Database Conference, pp. 144–154. Citeseer (1981)
19. Greenfield, P., et al.: Compensation is not enough. In: Proceedings of the 7th International Conference on Enterprise Distributed Object Computing, p. 232. IEEE Computer Society, Washington, DC (2003)
20. Holliday, M.A., et al.: A generalized timed Petri net model for performance analysis. IEEE Transactions on Software Engineering (12), 1297–1310 (1987)
21. Kramer, J., McGee, J.: Concurrency: State Models and Java Programs (1999)
22. Li, J., Zhu, H., Pu, G., He, J.: Looking into compensable transactions. In: 31st IEEE Software Engineering Workshop, SEW 2007, pp. 154–166. IEEE (2007)
23. Lin, L., Liu, F.: Compensation with dependency in web services compositions. In: International Conference on Next Generation Web Services Practices, NWeSP 2005, pp. 183–188. IEEE (2006)
24. Massuthe, P., Schmidt, K.: Operating guidelines-an automata-theoretic foundation for the service-oriented architecture. In: Fifth International Conference on Quality Software (QSIC 2005), pp. 452–457. IEEE (2005)
25. Papazoglou, M.P., Georgakopoulos, D.: Service-oriented computing. Communications of the ACM 46(10), 25–28 (2003)
26. Pires, P.F., et al.: Webtransact: A framework for specifying and coordinating reliable web services compositions (2002)
27. Potts, M., et al.: Business Transaction Protocol Primer. OASIS Committee Supporting Document (2002)
28. Reisig, W.: Simple Composition of Nets. In: Franceschinis, G., Wolf, K. (eds.) PETRI NETS 2009. LNCS, vol. 5606, pp. 23–42. Springer, Heidelberg (2009)
29. Vardi, M.: An Automata-Theoretic Approach to Linear Temporal Logic. In: Moller, F., Birtwistle, G. (eds.) Logics for Concurrency. LNCS, vol. 1043, pp. 238–266. Springer, Heidelberg (1996)

Model-Driven Development of Resource-Oriented Applications

Silvia Schreier
Supervised by: Bernd Krämer

University of Hagen, 58084 Hagen, Germany
silvia.schreier@fernuni-hagen.de

Abstract. Resource-oriented applications are based on the architectural style Representational State Transfer (REST). Current frameworks support the implementation phase of REST applications, but offer no provisions for the analysis and design task. In addition, there are at most informal guidelines to accommodate REST constraints. This thesis suggests the model-driven development of resource-oriented applications that facilitates the observation of REST constraints and provides the advantages of model-driven approaches like formalized and platform-independent design solutions or early validation options. The current state of research and future research steps will be presented.

Keywords: REST, resource-oriented applications, web services, model-driven development.

1 Introduction

Representational State Transfer (REST), an architectural style that was introduced by Roy Fielding in 2000 [6], gains more attention particularly in the context of developing web based applications. The most important implementation of REST is the World Wide Web with the Hypertext Transfer Protocol and its other standards. Based on this ideas RESTful services [15] have evolved. They are an alternative to web services based on remote procedure calls (RPC) [14].

Resource-oriented applications are server applications offering an interface consisting of linked *resources*. A *resource* is identifiable and can be manipulated by *representations* using a *uniform interface*, which is defined by the protocol. The form of a *representation* is defined by a *media type*.

There are many technical solutions and frameworks, e. g., Jersey [2] and Webmachine [3], that support the implementation of such applications. However, only parts of the development process are usually supported by (semi-)formal models, e. g., documentation [5] and composition [13]. Particularly, for the earlier phases, like analysis and design, no suitable models for resource-oriented applications exist.

Vinoski [19,21,22] remarks the missing tool support and the lack of best practices as open issues in the development of resource-oriented applications. Beside, he mentions that most programmers are used to programming languages which

G. Pallis et al. (Eds.): ICSOC 2011, LNCS 7221, pp. 201–206, 2012.

use specific interfaces instead of a uniform one and that this is why many developers have objections [20].

To solve these issues, model-driven development (MDD), a generic term for techniques for creating software automatically from formal models, seems to be an appropriate paradigm [17]. The main advantage of MDD is the higher level of abstraction achieved, which allows designers to document, discuss and validate designs and apply patterns and best practices in a uniform and technology-independent way. Furthermore, it frees the developers from error-prone and tedious routine work. Because expert knowledge can be incorporated into the code generators, the source code and software architecture is uniform and of high quality. Even after changes generated documentation is up to date which supports the evolvability of resource-oriented applications. The key question of our research is whether methodologies and techniques of MDD can help to enhance the development and the understanding of resource-oriented applications.

After analyzing the research challenge in the next section, Sect. 3 describes the research plan of this thesis. We conclude in Sect. 4 with a summary.

2 Research Challenge

For historical and other reasons, RPC-based services are still dominating service-oriented applications even in areas that are well suited for a resource-oriented approach. Pautasso et al. [14] identify a set of problems in their architectural decisions based comparison of REST with WS-* services. The confusion about best practices and the missing standardization of design methods are mentioned as important weaknesses [14]. This confusion is based on the different ways RESTful services are implemented and which REST constraints they fulfill.

A lot of practical guides for developing RESTful services [15] and several frameworks have arisen. But best practices are described in a natural language and do not offer a common vocabulary. Because it is hard to develop and to discuss a design without more formal foundations, the first challenge is to identify a vocabulary for the different development phases and - based on that - a suitable metamodel.

A secondary research question we want to investigate is whether such a vocabulary can foster a better understanding of the foundations of resource-oriented, in contrast to operation-centric or object-oriented, design. Some existing work in the field of MDD for RESTful services has a too operation-centric view and therefore does not support a better understanding of the uniform interface [18]. The work on transformations from an operation-centric to a data-centric view of Laitkorpi et al. [11] should be taken into account when researching if such a metamodel can support the mapping from the application domain to an analysis model. The authors present a model-driven process for designing RESTful services, but skipped code generation. Instead of building a metamodel, which provides guidance, they base the identification of *resources* and the domain analysis on models and model transformation using the Unified Modeling Language (UML) and the Web Application Description Language (WADL) [8].

The existing implementation frameworks exhibit disadvantages because developers have to overcome the dichotomy of a resource-oriented application perspective and languages that promote specific interfaces and a more operation-centric perspective [20]. Furthermore, in the case of many frameworks the source code seems to contain several code duplicates, which typically causes copy and paste errors. This assumption should be explored in more detail by analyzing different implementations in different languages. If this holds true, it should be shown if code generation is possible and to which extent.

Particularly, in the field of documenting resource-oriented applications a lot of research has been done. Kopecký et al. [10] suggest microformats, which contain a model for RESTful services, but focus on documentation and discovery. With the same focus and also for composition Alarcón and Wilde [5] introduce a metamodel which is the basis for the Resource Linking Language (ReLL). The most important difference to WADL [8] and the microformats is that links are first class citizens in ReLL. Because currently the documentation has to be written manually, evolution often causes inconsistencies. To minimize the effort for creating and maintaining the documentation, the question if a (semi-)automatic generation for documentation is possible has to be answered.

Furthermore, the missing possibility to generate client stub code for RESTful applications is mentioned as another disadvantage of RESTful services by Pautasso et al. [14]. Another challenge is to show that it is possible to generate (parts of) client stubs based on a (semi-)formal model. A common use case is that existing legacy systems shall be extended by a resource-oriented interface. So the linking of such systems is another challenge. Having a formal metamodel, provides the opportunity to verify if a model satisfies desired properties. For example, it is necessary that the resources are linked to support hypermedia. Hence, it would be desirable that the graph built of all *resource types* and their links in between is (strongly) connected. Another interesting question is what properties can be verified with such a metamodel.

3 Research Plan

The first step on the research agenda is the development and evaluation of a metamodel. After finishing a first stable version, a textual and visual language for a better usability are planned as well as code, documentation, and client generation for different languages, frameworks, and formats. With these results a MDD of resource-oriented applications becomes possible. In addition to the tools, a development process needs to be defined for different scenarios, e. g., developing a new application or designing an interface for a legacy system. The approach and the prototypical implementations of the tools based on the Eclipse project [1] shall be tested with different case studies.

3.1 Defining a Metamodel

One of the most important aspects of this thesis is the development of the metamodel because its expressiveness determines the field of application addressed

and the benefits of the metamodel. The version presented in [16] needs further refinements. Additionally, constraints and default values [17] need to be included.

To be able to model the functional aspects of a resource-oriented application for the entire development process, not only the structure but also the behavior needs to be taken into account [16]. So the metamodel consists of two parts, one for describing the structure, defining the *resource types* and the links in between, and one for describing the behavior. We suggest to use state machines to model the resource states and which methods are supported in each one. To describe the behavior of one method in more detail, imperative instructions are used.

The biggest challenges in this part is finding the proper elements and a suitable vocabulary on the one hand; on the other hand, we need to model *resource* manipulations that go beyond create, read, update and delete.

After the refinements the metamodel shall be applied to different case studies and compared to other models in this context [24]. Furthermore, it needs to be explored if one model can cover the requirements of all phases or if different models with transformations inbetween are more suitable.

Based on the chosen tools an automatic generation of a tree view editor is possible. Using such an editor is helpful and an alternative to describing the model using plain XML in a text editor. But a language with a concrete syntax which was designed for that domain is even better. This requirement leads to the next step of developing a textual and visual language.

3.2 Development of Textual and Visual Language

A metamodel is only the abstract basis for a framework for MDD. To get a useful domain-specific language (DSL) [7], only the concrete syntax needs to be added. Stahl et al. [17] mention that visual languages are better in illustrating structures and relationships. Additionally, they offer a better basis for discussions with bigger groups. If too many details are contained, a diagram becomes confusing for the reader. But for modeling a real world application the details need to be described as well. With textual languages defining details is more efficient. Because developers have different preferences regarding the kind of language, the goal is to develop both language types for this metamodel. For instance, the visual language could be used for the first analysis and identification of the different *resource types* and their linking as well as for showing the behavior of a *resource type* in terms of state machines.

The languages can be evaluated by testing their suitability for the documentation of best practices and patterns.

For developing easy to use languages, best practices and quality criteria for designing textual [7,23] and visual [12] languages should be taken into account.

3.3 Code Generation

As mentioned before, there are several frameworks in different languages for implementing resource-oriented applications. But these languages do not have

resources or *resource types* as first class citizens. That is, why the different paradigms and abstractions become mixed.

If the metamodel allows us to build detailed models, we can generate code for different frameworks to reduce the amount of manually written code duplicates and to follow the *don't repeat yourself* (DRY) principle [9]. Probably, it is not possible to generate all necessary code, but avoiding repeating yourself is desirable. It is necessary to find the tradeoff between a model that overly detailed is hard to understand and a model that allows as much code generation as possible. The goal for this step is to provide code generators for at least one object-oriented framework, e. g., Jersey [2], and for one written in a functional language, e. g., the Erlang toolkit Webmachine [3]. The biggest challenge in code generation will be the different paradigms. This will show if the designed metamodel is independent enough from these paradigms.

Until now the concrete appearance of representations is not part of the model and only the contained data are described. For solving this issue, templates describing the concrete syntax can be used for representations sent by the server. For retrieving data from the receiving representations, a support for at least widely spread formats like XML and JSON should be developed to reduce the amount of manually written code again. For XML, e. g., XPath [4] could be used, similar to the selectors suggested in [5].

3.4 Documentation and Client Generation

There are several alternatives for documenting resource-oriented applications all having their advantages and disadvantages. The goal is to develop model transformations which allow us to generate WADL and ReLL documentations or at least a part of it out of the model. A first comparison of the current metamodel and the metamodels of WADL and ReLL shows that corresponding elements can be found. Hence, the development of such a transformation seems to be promising. The generation of (parts of) client stubs for different languages could be based on the documentation in WADL or ReLL to achieve a loose coupling from the server implementation.

4 Conclusion

Even though there is a lot of existing related work in developing resource-oriented applications no model-driven approach for the entire process exists. The goal of this thesis is to develop a metamodel for resource-oriented applications. Based on this theoretical work a framework for MDD of such applications is planned including the support for analysis and design as well as implementation and documentation. In case this can be achieved, it will be established that a metamodel can support the entire MDD process and what benefits and drawbacks it has. In some parts existing models can be used for comparison. Additionally, this thesis can benefit from a lot of best practices and tools in the field of developing frameworks for MDD.

References

1. Eclipse, http://www.eclipse.org/
2. Jersey, JAX-RS (JSR 311) Reference Implementation, http://jersey.java.net/
3. Webmachine, http://webmachine.basho.com/
4. XML Path Language (XPath) 2.0, http://www.w3.org/TR/xpath20/
5. Alarcón, R., Wilde, E.: RESTler: crawling RESTful services. In: Proceedings of the 19th International Conference on World Wide Web, pp. 1051–1052. ACM (2010)
6. Fielding, R.T.: Architectural Styles and the Design of Network-based Software Architectures. Ph.D. thesis, University of California, Irvine (2000)
7. Fowler, M.: Domain-specific languages. Addison-Wesley Professional (2010)
8. Hadley, M.: Web Application Description Language. World Wide Web Consortium Member Submission SUBM-wadl-20090831 (August 2009), http://www.w3.org/Submission/2009/SUBM-wadl-20090831/
9. Hunt, A., Thomas, D.: The Programatic Programmer. From journeyman to master. Addison-Wesley (1999)
10. Kopecký, J., Gomadam, K., Vitvar, T.: hRESTS: An HTML Microformat for Describing RESTful Web Services. In: 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, pp. 619–625. IEEE (2008)
11. Laitkorpi, M., Selonen, P., Systa, T.: Towards a Model-Driven Process for Designing ReSTful Web Services. In: 2009 IEEE International Conference on Web Services, pp. 173–180. IEEE Computer Society (2009)
12. Moody, D.L.: The "Physics" of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering. IEEE Transactions on Software Engineering 35, 756–779 (2009)
13. Pautasso, C.: Composing RESTful Services with JOpera. In: Bergel, A., Fabry, J. (eds.) SC 2009. LNCS, vol. 5634, pp. 142–159. Springer, Heidelberg (2009)
14. Pautasso, C., Zimmermann, O., Leymann, F.: Restful Web Services vs. "Big" web services: Making the Right Architectural Decision. In: Proceedings of the 17th International Conference on World Wide Web, pp. 805–814. ACM (2008)
15. Richardson, L., Ruby, S.: RESTful Web Services. O'Reilly Media (2007)
16. Schreier, S.: Modeling RESTful applications. In: Proceedings of the Second International Workshop on RESTful Design, pp. 15–21. ACM (2011)
17. Stahl, T., Völter, M., Efftinge, S., Haase, A.: Modellgetriebene Softwareentwicklung: Techniken, Engineering, Management. dpunkt.verlag (2007)
18. Valverde, F., Pastor, O.: Dealing with REST Services in Model-driven Web Engineering Methods. In: V Jornadas Científico-Técnicas en Servicios Web y SOA, JSWEB 2009 (2009)
19. Vinoski, S.: REST Eye for the SOA Guy. IEEE Internet Computing 11(1), 82–84 (2007)
20. Vinoski, S.: Demystifying RESTful Data Coupling. IEEE Internet Computing 12(2), 87–90 (2008)
21. Vinoski, S.: RESTful Web Services Development Checklist. IEEE Internet Computing 12(6), 94–96 (2008)
22. Vinoski, S.: RPC and REST: Dilemma, Disruption, and Displacement. IEEE Internet Computing 12(5), 92–95 (2008)
23. Völter, M.: MD*/DSL Best Practices (2011), http://voelter.de/data/pub/DSLBestPractices-2011Update.pdf
24. Wilde, E., Pautasso, C. (eds.): REST: From Research to Practice. Springer (2011)

Human Task Management for RESTful Services

Daniel Schulte
Supervised by : Bernd J. Krämer

FernUniversität in Hagen, 58084 Hagen, Germany
Daniel.Schulte@FernUni-Hagen.de

Abstract. Human task management is an integral part of WS-* based service-oriented solutions like ESBs as human interactions are important for business processes. Also alternative REST-based service-oriented solutions provide support for business processes but lack interoperable human task management solutions.

In the PhD project we are investigating possibilities and limits of incorporating human tasks in REST-based service solutions. We present a first approach and discuss the design of the remaining research process including challenges, potential benefits, and open issues.

Keywords: Human Task Management, REST, Hypermedia.

1 Introduction

Business process engines automate business processes but do also require user interactions to perform not automatable tasks [6]. Common standards like BPEL solve the process automation for web services. WS-HumanTask and BPEL4People specify additional components for user interactions. But these solutions focus on established enterprise IT environments utilizing WS-* technology stacks.

An alternative service approach relies on REST [2,8] and utilizes the web technology stack to provide its services. Whereas RESTful service composition is addressed in recent research (e. g., JOpera, see [9]), human task management for RESTful services and web applications is not examined, although many web applications already contain human tasks (e. g., conference management systems) or support their execution (e. g., online office applications). Instead, proprietary portals and e-mail communication are used regularly to notify users that tasks are available to work off. Proprietary portals for task management lead to isolated solutions and applications for e-mails lack task management facilities.

To overcome the weaknesses of these stopgap solutions, we aim to find a systematic and easy to use human task management fitting in a web context. Our research investigates axioms and constraints guiding the architecture design of web applications (providing RESTful services) that will enable users to oversee and manage their tasks independently of applications involved and their vendors.

Section 2 discusses challenges for web-scale human task management as well as current approaches and introduces our research hypothesis. Section 3 identifies main components of our proposed solution and depicts further challenges. Section 4 concludes this paper naming expected impacts.

G. Pallis et al. (Eds.): ICSOC 2011, LNCS 7221, pp. 207–212, 2012.

2 Web-Scale Human Task Management

Human tasks are —broadly defined— actions that are carried out by humans. This spans simple data entry tasks controlled by rigid input masks but also creative tasks of knowledge workers.

The management of tasks comprises their collection, the maintenance of metadata like deadlines, the view and manipulation of these metadata by users as well as scheduling of tasks, and so on. When task-aware applications provide task descriptions incl. metadata to affected humans, task management can be (partially) automated.

Challenges for an (automated) web-scale human task management are discussed in sec. 2.1, related solutions and approaches are presented in sec. 2.2, and sec. 2.3 introduces the research hypothesis underlying our approach to web-scale human task management.

2.1 Challenges for Web-Scale Human Task Management

To survey and manage tasks efficiently, users should be able to access all their tasks by a *single worklist* and have access to *task management facilities* that can handle tasks regardless of the origin. Ultimately, users need only one task management application (not one task management application per organization, (virtual) enterprise, team, task provision application, etc.). To realize this vision in the web, some requirements have to be considered [12]:

- Task announcement automation: As tasks emerge in task-aware applications, these applications should *add tasks to users worklists* incl. important metadata like deadlines directly. As tasks change over time (state, deadlines, etc.), they should also *update these metadata* automatically.
- Autonomy of task execution: Human tasks may originate in various contexts with different execution steps, states, and security requirements. They are already contained in existing applications. Therefore, the task execution should be controlled by these self-contained applications.
- Autonomy of group management: The assignment of tasks within teams may depend on current work loads, on team member roles, group policies or on some other kind of agreement. To support individual assignment patterns, independent group management solutions are needed.
- Autonomy of task management: Respecting the freedom of users means that they should be in control of their own tasks, which includes the delegation and deletion of tasks without restrictions due to arbitrary web application.
- Flexibility of tasks lifecycle and model: Task types may span routine jobs, adaptive tasks and innovative ones [3], and may be executed by one or several persons cooperatively or competitively, to name a few options. Therefore, no common task lifecycle or model can be expected. Task management has to be flexible to support different not a priori known task lifecycles and models.
- Interoperability: As arbitrary task-aware web applications should provide task descriptions, interoperability between these applications and task management applications is crucial.

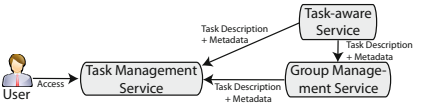


Fig. 1. Service Overview

Furthermore, web applications evolve independently, may be substituted or temporary offline. Human task management applications must handle these inherent characteristics of the web but may also benefit from its features like common authentication and authorization mechanism based on OpenID and OAuth.

Core services are depicted in fig. 1: A task management service allows a user to access and manage her tasks, task-aware services provide information about outstanding tasks and may support the task execution, and group management services enable one to consider group structures or the like for task assignment.

2.2 Approaches to Human Task Management

Existing task-aware web applications often use e-mail —a distributed, interoperable system with user-centered access possibilities— for task management although task management functions are absent. Therefore, several approaches try to extend it. Whittaker and Sidner propose to redesign e-mail for task management using threading and semantic clustering of mails and marking mails that require action together with the option to program reminders [13]. Whittaker, Bellotti & Gwizdka propose a combination of centralization (all personal information management should be done in email programs) with explicit PIM functions and information extraction [14]. Li et al. propose agents to process ingoing, outgoing, and existing e-mails for automation of manual work by intelligent applications [7] which can be adapted to extract task descriptions and metadata from e-mails to offer a worklist.

However, this approaches support, e. g., automated task description update only partial and separate task management from the web as they believe, that "it seems highly unlikely that users will abandon email for dedicated task-management tools because such tools fail to support the important reminding aspects of task management" [13]. But new browser plug-ins, notifications on smartphones and other developments improved the situation and can —as users use several devices in parallel— benefit from web managed task descriptions.

Another already mentioned approach originates in the WS-BPEL community: WS-HumanTask [1] specifies a human task concept with a detailed state and transaction system for tasks. It focuses on tasks executed by one person and is build on the WS-* stack. Questions regarding, e. g., distribution are not addressed. Task access and group management is solved proprietarily. Thus, it is appropriate for usage in enterprises with a distinct scope where IT departments manage deployment and people assignment but not applicable for a web-scale human task management.

Collaboration integration approaches like the action-centric HERMES [5] provide deep integrations at the expense of autonomy of applications (e. g., shared internal data) but do also not consider distribution and restrict the autonomy of integrated services.

Altogether, a web-scale human task management solution considering all named requirements and challenges is yet not established.

2.3 Research Hypothesis and Research Plan

As task-aware web applications rely on the web technology stack, we propose a solution that also utilizes this stack. As REST's constraints guided its development and fosters, among others, simplicity, independent evolvability, scalability and extensibility [2], it should also guided the development of a web-scale human task management solution.

Furthermore, "[t]he Web is intended to be an Internet-scale distributed hypermedia system" [2]. Hypertext links are a defining characteristic of the web and the simplicity of creating links "is partly (perhaps largely) responsible for the success of the hypertext Web as we know it today" [4]. Links allow the interconnection of information across multiple organizational boundaries and foster loose coupling and independent evolvability.

Automatically created worklists with some task metadata can link to task-aware web application, so that user can easly access them. This respects the autonomy of involved applications, and enables flexible tasks lifecycles and models as task management applications need no deep knowledge about tasks.

Nowadays, HTML and JavaScript provide not only navigable hyperlinks but also interactive controls via, e. g., HTMLs forms and JavaScipts XMLHttpRequests. Thus, task descriptions and metadata can be enriched with powerful application control information to enable even advanced task specific management facilities by hypermedia controls and code on demand.

Hence, our research hypothesis is that we can establish a web-scale human task management respecting REST constraints and making extensive use of its hypermedia as the engine of application state constraint. But to achieve interoperability and automate, e. g., the collection and update of task descriptions, some additional constraints have to be observed. Therefore, our research will

- investigate essential constraints for web-scale human task management,
- demonstrate its suitability in case studies with a prototype solution of a REST-conform human task management solution, and
- evaluate the solution using common workflow resource patterns, such as those patterns identified in [10] and already used to evaluate BPEL4People and WS-HumanTask [11].

3 Components and Challenges of a RESTful Solution

Figure 1 depicted a high level view on the task management problem domain which is further decomposed in fig. 2 and identifies the main components and resources of a RESTful solution.

A *task list* comprises all tasks of a user. As a resource, it is equipped with an URI and `POST`ing a task description to it will (a) add it to this list and (b) create a *task description* subresource. This subresource is equipped with an URI

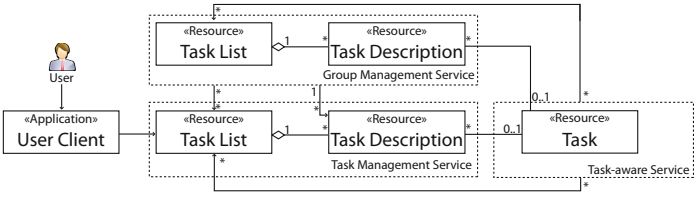


Fig. 2. Component Overview

itself which will be returned to the adding application for subsequent updates (by `PUT`ting the updated task descriptions to this subresource). Task descriptions contain metadata about tasks as well as application control information for advanced task specific management facilities.

If the execution of the task is supported by a web application, a *task* resource will be created which can be linked in the task description and invoked by a client (e. g., in a browser). And a *group management* can be realized as intermediary acting as a task management component for task-aware web applications and vice versa.

Consequently, the information flow between task management component and task-aware web application is modeled as a push architecture to reduce the workload of the task-aware web applications supporting scalability (a message exchange only occurs when an update is available), and to reduce latency of updates supporting accuracy (the update occurs as soon as an update is available).

On this technical level, further challenges need to be addressed, for instance, how to specify and describe the infrastructural parts to ensure interoperability but also independent evolvability of components. Typically, is should be based on media types and link relations to foster exploration of web applications and extensibility and should consider already established media types describing tasks as well as characteristics of the web infrastructure like caching for scalability. As the web includes malicious parties and task descriptions should only be modifiable by originators and its commissaries, they need to be secured.

Web applications which do not explicitly model tasks, like Google Docs, need to be made task-aware without changing their implementation. Other web applications which rely on e-mail to inform users about outstanding tasks need to be adapted.

A first task management component prototype is developed as Java application utilizing Jersey. An android client and a PHP based web client can access the managed tasks. A framework for using currently task-unaware web applications as task-aware ones noninvasively is under development.

4 Conclusion

Tasks of users in the web are spread over different web applications offered by independent organizations. Therefore, web-scale human task management should help users to overview and manage their tasks efficiently and independently and

has —compared to other task management scopes— some unique characteristics, in particular the distribution of task-related information and the autonomy of task-aware web applications, group management applications and task management applications. So far, no web-scale human task management is established. Our approach to this problem bases on the REST architecture style, the web technology stack and heavily relies on hypermedia.

Several challenges concerning this approach are introduced and —as the overall suitability of it— have to be studied in the future. But the expected outcomes are not restricted to the problem domain of task management as a better understanding of interoperability and evolvability may promote the interconnection of applications for other domains, too. The outcomes should further facilitate reusability and adaptability of web applications and processes in unforeseen ways.

References

1. Agrawal, A., et al.: Web Services Human Task (WS-HumanTask), Version 1.0. (2007)
2. Fielding, R.T.: Architectural Styles and the Design of Network-based Software Architectures. PhD thesis, University of California, Irvine (2000)
3. Hoffmann, F.: Aufgabe (german). In: Grochla, E. (ed.) Handwörterbuch der Organisation, Poeschel, Stuttgart, pp. 200–207 (1980)
4. Jacobs, I., Walsh, N.: Architecture of the World Wide Web, Volume One : W3C Recommendation (December 15, 2004). http://www.w3.org/TR/2004/REC-webarch-20041215/ (August 18, 2011)
5. Kapos, G.-D., Tsalgatidou, A., Nikolaidou, M.: A Web Service-Based Platform for CSCW over Heterogeneous End-User Applications. In: PDCS 2004, pp. 462–469 (2004)
6. Kloppmann, M., et al.: WS-BPEL Extension for People – BPEL4People. IBM & SAP (2005)
7. Li, W., Zhong, N., Yao, Y., Liu, J.: An Operable Email Based Intelligent Personal Assistant. World Wide Web 12(2), 125–147 (2009)
8. Pautasso, C., Zimmermann, O., Leymann, F.: RESTful web services vs. "big" web services: making the right architectural decision. In: WWW 2008, pp. 805–814 (2008)
9. Pautasso, C.: Composing RESTful Services with JOpera. In: Bergel, A., Fabry, J. (eds.) SC 2009. LNCS, vol. 5634, pp. 142–159. Springer, Heidelberg (2009)
10. Russell, N., ter Hofstede, A.H.M., Edmond, D., van der Aalst, W.M.P.: Workflow Resource Patterns. Eindhoven University of Technology (2004)
11. 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. BPM Center (2007)
12. Schulte, D.: Web-Scale Human Task Management. In: Crnkovic, I., Gruhn, V., Book, M. (eds.) ECSA 2011. LNCS, vol. 6903, pp. 190–193. Springer, Heidelberg (2011)
13. Whittaker, S., Sidner, C.: Email Overload: Exploring Personal Information Management of Email. In: CHI 1996, pp. 276–283 (1996)
14. Whittaker, S., Bellotti, V., Gwizdka, J.: Email in personal information management. Communications of the ACM 49(1), 68–73 (2006)

CLAM: Cross-Layer Adaptation Management in Service-Based Systems*

Asli Zengin
Supervised by: Marco Pistore

Fondazione Bruno Kessler – IRST, Trento, Italy
{zengin,pistore}@fbk.eu

Abstract. Service-based systems (SBS) have a complex layered structure where the service-based application (SBA) is implemented through a composition of services, which run on top of infrastructures. Adaptation is not straightforward when we take into account the heterogeneous and dynamic execution context of such complex systems. While several state-of-the-art approaches, unaware from each other, target different problems at specific parts of the system, the isolated enactment of those adaptations results in ignoring the overall impact on the whole SBS. In this ongoing PHD work, we propose a complete, flexible and extensible solution for the cross-layer adaptation of SBSs. Our proposed solution integrates and coordinates existing analysis and adaptation tools to assess the impact of an adaptation at different system levels. Moreover, throughout the impact analysis, starting from an initial adaptation trigger it incrementally constructs adaptation strategies, consistent with the overall system.

1 Introduction

Service-based systems must provide their functionality with the required/agreed qualities of services, cope with the unreliability of the infrastructure on which they operate, and also deal with the changes in the context in which they are executed, or in the partner services with which they interact. This means that all these problems must be discovered as soon as they appear, and these systems must be able to adapt their behavior to handle them.

Adapting the behavior may mean changing the actual composition of services, or selecting different partner services, but the satisfaction of the service level agreements in place may also impose changes in the way services are offered. For example the supervision system may re-negotiate some quality parameters with the providers of the partner services, change the configuration of the engine that runs the composition (e.g. BPEL process), and even adjust the infrastructure (resources) used by the application and its partners.

Many existing solutions [1,2,3,4,5] have addressed adaptation in a "local" way by only considering one concern at one layer. However, if we considered the whole SBS stack, some adaptations may trigger others, or they may influence some quality parameters, or even the operation, or parts of the system. Adaptation is thus a cross- and

* 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. Pallis et al. (Eds.): ICSOC 2011, LNCS 7221, pp. 213–219, 2012.

multi-layer problem and we must take into account the entire system consistency rather than only local problems and their specific solutions. To address this issue, this PHD work introduces a novel approach that comprises a comprehensive cross-layer system representation, and an extensible platform that integrates and coordinates existing analysis and adaptation tools in a holistic way. Our approach assesses the impact of an adaptation at different system levels, and in case of detecting an adaptation's incompatibility, it proposes new adaptations and incrementally constructs adaptation strategies (a set of adaptations), consistent with the whole system.

2 Problem Statement

We present a scenario to illustrate the cross-layer adaptation problem. We define the SBS layers for the scenario as follows: *(i) Application layer:* "Call & Pay Taxi" composite service (CPTS), implemented as a BPEL process. *(ii) Service layer:* The partner services of our application, namely a short messaging service (SMS), a location service (LS) and a payment service (PS) provided by the telecom company, and the taxi service (TS) provided by the taxi company. *(iii) Infrastructure layer:* The underlying platforms on top of which CPTS, SMS, LS, PS and TS run.

In CPTS, the client requests a taxi by sending a text message (SMS) to the application. Then, his/her location is identified and the taxi company is contacted to organize the real taxi service. After transporting the user to the destination, the process terminates with a successful payment.

Let us consider an adaptation case on our scenario: The CPTS provider decides to switch to a cheaper telecom provider. This means replacing SMS, LS and PS in the BPEL process. However, there is a problem with the new LS service's output message format. It provides client's location in geographical coordinates instead of full address while in our application design we use full address as the input message to the TS. To remove this new data mismatch, the service composition is adapted by adding a data mediator service in the workflow. Yet it triggers a new problem: We notice that the new service for data mediation is too costly and in fact increases the overall cost of the process in an unforeseen way, eventually conflicts with the initial adaptation goal, which is cost reduction.

Problem. With the existing approaches when an adaptation is performed, it targets a particular problem occurring at a specific aspect of a specific SBS layer. Thus, they tend to propose local solutions to local problems in a way that is isolated from the overall application context. To avoid this, we must understand the impact of a change across different system layers, which have their own characteristics and constraints. In our example while we are trying to improve the cost, yet we do not know the consequences of replacing the services for the whole system.

We need a holistic approach with the following research challenges in hand [6]: (i) Address adaptation problem globally for the SBS. (ii) Provide a neat and comprehensive reference model for system aspects and layers and the adaptations in the SBS. (iii) Integrate existing and forthcoming solutions that are in isolation into a consistent and coherent solution. (iv) Understand the impact and consequences of possible adaptations for all the system elements. (v) Release an innovative and extensible cross-layer

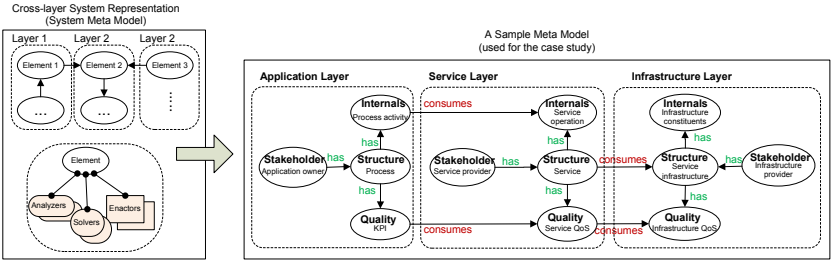


Fig. 1. System Modeling for CLAM

adaptation solution to devise efficient adaptation plans consistent with the overall SBS. (vi) Evaluate the proposed solution on various application domains.

3 Proposed Solution

Our approach exploits two key facts about SBSs to solve the cross-layer adaptation problem: First, SBSs already contain the dependencies among different layer elements implicitly while those dependencies are not trivial among adaptations in different layers. Hence, we would like to benefit from the layer dependencies known for the system, and model the SBS in a cross-layer manner so that the dependencies get explicit to be easily used by our technique. Second, there are several state-of-the-art approaches for various types of SBS analysis and adaptation, and they already run in the system in an uncoordinated way. In our approach we would like to plug those mechanisms in a cross-layer adaptation management (CLAM) platform so that we can reuse them to analyze the impact of an adaptation trigger and to extend it if required. Thus, CLAM aims to coordinate the current approaches to prevent conflicting adaptations and to produce a final, validated adaptation strategy aligned with the overall SBS. Taking into account these facts, our solution has three bases:

1) Cross-layer System Representation. We solve cross-layer adaptation problem on the meta model of the system. It is basically a directed graph where nodes represent the system elements from different layers and the edges represent the relations between these elements. Apart from system elements and their relationships, we have the system artifacts associated to the elements. They are the analysis and adaptation mechanisms that are available for the SBS: *(i) Analyzers* check the compatibility of a new adaptation for a system element that they are associated to. *(ii) Solvers* get an incompatibility problem triggered by an analyzer and try to propose an adaptation to handle the problem. *(iii) Enactors* implement final adaptation strategies validated by CLAM.

The meta model for our case study could be found in Figure 1. While this model is created for the application scenario in this work, meta models for other application domains could involve different system elements and relationships. Our approach is able to work with various application domains, as long as the application system can be be modeled by its elements and their dependencies to each other.

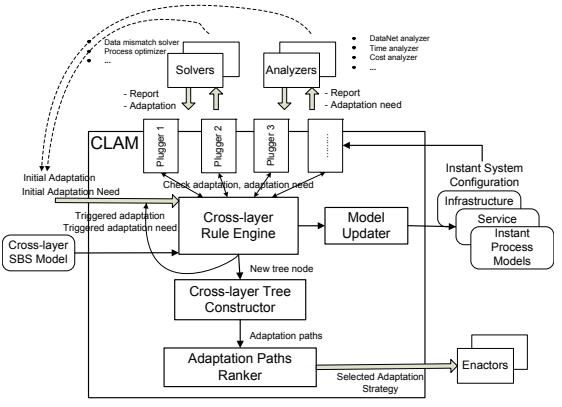


Fig. 2. CLAM: Cross-layer Adaptation Manager

2) Methodology. We perform the entire adaptation analysis as a continuous execution of predefined *rules*. These rules determine how to navigate the SBS meta model to identify the system elements affected by an adaptation, and subsequently how to decide which analyzers and solvers to invoke, and finally how to gradually construct a cross-layer adaptation tree upon receiving results from those external tools. Cross-layer adaptation tree is basically the output of the impact analysis where the branches correspond to alternative adaptation strategies that can overall address the negative impacts of an initial adaptation. Figure 3 shows a sample tree produced by our approach for the analysis of the adaptation case study presented in Section 2.

3) Supporting Tool. The architecture for CLAM platform is given in Figure 2. In order to perform a comprehensive impact analysis for an adaptation, CLAM executes the rules defined in the methodology. Moreover, through its pluggers it provides an integration platform where one can plug in new analyzers and solvers, and improve the analysis by including more artifacts in the system.

We have implemented a first version of CLAM where we have a time and cost analyzer associated to the KPI node in the meta model, then DataNet analyzer associated to the process activity node, finally process optimizer and data mismatch solver associated to the process node. All of them are real tools from the state-of-the-art.

In this first implementation, let us see how CLAM platform works to produce the tree in Figure 3 for our case study: CLAM makes use of instantiations (instant model, e.g. M_0, M_1...) of the meta model with concrete value assignments of system elements to keep track of alternative adaptation proposals during an analysis. When the new adaptation trigger "replace telecom provider X by Y" comes to the rule engine, the engine identifies which nodes in the instant model are changed (service provider, service, service operation, service QoS) and which nodes might be affected due to those changes (process activity, KPI). For the changed and affected nodes it identifies the associated artifacts (datanet, cost and time analyzers) to be invoked for the impact analysis. Then, it contacts the model updater to update the instant model with the changes. On the new instant model it initiates the analysis by calling the analyzers one by one. If they report

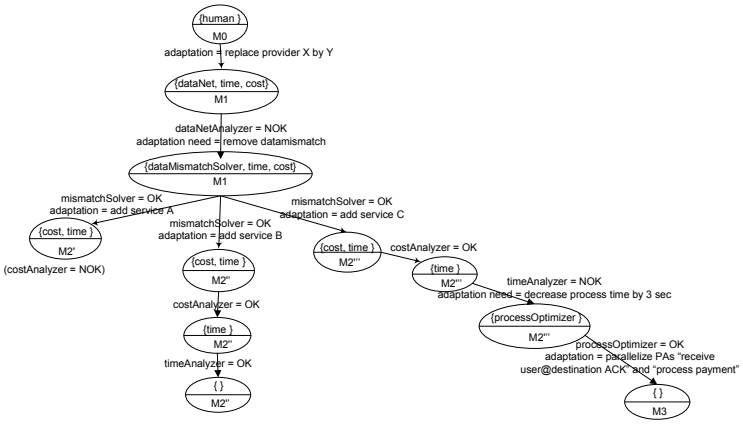


Fig. 3. CLAM Tree for Alternative Adaptation Paths

back some problems (adaptation needs), then it determines and contacts a proper solver to solve the problem newly occured. In this way, it continues the analysis until all the necessary artifacts are called.

The outcome is the cross-layer adaptation tree with some alternative adaptation paths (strategies). We take a tree branch as an alternative strategy only if all the problems on the path have been solved properly by the artifacts. Finally based on some criteria (e.g. shortest path, minimum cost path etc) we can select one of the paths, and through enactors we can deploy the adaptations present on the path.

Preliminary Results. The construction of the tree in Figure 3 took around 3 seconds on the 2 GHz Intel Pentium M Processor Windows XP laptop with 1 GB of RAM. Our experiences from the first implementation show that it is worth investigating further the on-line usage of our solution considering the time performance. Moreover, our approach is light-weight and extensible to plug in various artifacts in CLAM.

4 Related Work

Our expected research impact with respect to the state-of-the-art is in two ways: First, existing adaptation work mostly focuses on specific aspects of the SBA where adaptation problem is solved merely in this narrow scope without taking into account its consequences for the whole SBS stack. The approaches in BPM adaptation [1], self adaptation and self healing systems [2], QoS-awareness [3], mediator design for service interactions [4] and finally context-awareness [5] are the prevalent ones in this category. Adaptations proposed by such approaches may be conflicting with each other, even they may be harmful for the application. Our solution aims to align and coordinate them effectively to prevent such cases.

Second, there are few approaches in literature that use cross-relations for adaptation. However, compared to those works our approach brings the novelty of extensibility and genericness, i.e., not being domain specific. In this category, we have [7] that analyzes the dependencies of KPI violations on quality metrics from different layers of an SBS. Then, [8] proposes a technique where the designer prepares the taxonomies of adaptation mismatches, and later designs the adaptation templates that define generic solutions to tackle mismatches. Finally, [9,10,11] have a cross-layer representation of the system. While [10,11] target limited number of adaptation cases such as service replacement, [9] uses the cross-layer model for monitoring and analysis rather than adaptation.

5 Conclusions and Future Work

We have presented a holistic approach to solve the cross-layer adaptation problem. In the remaining six months of this PHD work, we will work on the formalization of both the research problem and the proposed solution. This will directly contribute to the enhancement of the implementation. We will use more adaptation case studies and more application domains where we have different system meta models. In this way, we are planning to consolidate the platform regarding its extensibility and genericness. Meanwhile we will also improve the algorithm especially considering the tree construction to avoid infinite trigger of adaptations, i.e., to ensure the termination of the algorithm. Eventually, we would like to evaluate our approach in two ways: (i) investigating the usability of our solution at run-time, (ii) comparing two cases where in one case we run the adaptation and analysis tools coordinated by our approach, and in the other case we run them in an uncoordinated manner.

References

1. Brogi, A., Popescu, R.: Automated Generation of BPEL Adapters. In: Dan, A., Lamersdorf, W. (eds.) ICSOC 2006. LNCS, vol. 4294, pp. 27–39. Springer, Heidelberg (2006)
2. Charfi, A., Dinkelaker, T., Mezini, M.: A plug-in architecture for self-adaptive web service compositions. In: ICWS 2009: Proceedings of the 2009 IEEE International Conference on Web Services, pp. 35–42. IEEE Computer Society, Washington, DC (2009)
3. Christos, K., Vassilakis, C., Rouvas, E., Georgiadis, P.: Qos-driven adaptation of bpel scenario execution. In: ICWS 2009: Proceedings of the 2009 IEEE International Conference on Web Services, pp. 271–278. IEEE Computer Society, Washington, DC (2009)
4. Kongdenfha, W., Motahari-Nezhad, H., Benatallah, B., Casati, F., Saint-Paul, R.: Mismatch patterns and adaptation aspects: A foundation for rapid development of web service adapters. IEEE Transactions on Services Computing, 94–107 (2009)
5. Ardagna, D., Comuzzi, M., Mussi, E., Pernici, B., Plebani, P.: Paws: A framework for executing adaptive web-service processes. IEEE Softw. 24, 39–46 (2007)
6. Zengin, A., Kazhamiakin, R., Pistore, M.: Clam: Cross-layer management of adaptation decisions for service-based applications. In: 2011 IEEE International Conference on Web Services, pp. 698–699. IEEE (2011)
7. Kazhamiakin, R., Wetzstein, B., Karastoyanova, D., Pistore, M., Leymann, F.: Adaptation of Service-Based Applications Based on Process Quality Factor Analysis. In: Dan, A., Gittler, F., Toumani, F. (eds.) ICSOC/ServiceWave 2009. LNCS, vol. 6275, pp. 395–404. Springer, Heidelberg (2010)

8. Popescu, R., Staikopoulos, A., Liu, P., Brogi, A., Clarke, S.: Taxonomy-driven adaptation of multi-layer applications using templates. In: SASO 2010, pp. 213–222 (2010)
9. Baresi, L., Caporuscio, M., Ghezzi, C., Guinea, S.: Model-Driven Management of Services. In: ECOWS 2010, pp. 147–154 (2010)
10. Tripathi, U., Hinkelmann, K., Feldkamp, D.: Life cycle for change management in business processes using semantic technologies. Journal of Computers 3, 24 (2008)
11. Burgstaller, B., Dhungana, D., Franch, X., Grunbacher, P., López, L., Marco, J., Oriol, M.: Stockhammer: Monitoring and Adaptation of Service-oriented Systems with Goal and Variability Models. Technical report, Universitat Politècnica de Catalunya (2008)

Investigating Service-Oriented Business Architecture

Elisah Lemey
Supervised by: Geert Poels

Center for Service Intelligence,
Faculty of Economics and Business Administration,
Ghent University,
Tweekerkenstraat 2, 9000 Gent, Belgium
Elisah.Lemey@UGent.be

1 Context and Problem Statement

Enterprise Architecture (EA) is a term used in Information Management to refer to the joint design of an organization's business infrastructure and the information technology (IT) infrastructure. EA is a major instrument to achieve business/IT alignment, i.e. the strategic and operational integration of an organization's business and IT domains, which in turn contributes to effective enterprise governance of IT, i.e. the responsible investment in and management of IT resources such that IT delivers business value.

Organizations' IT infrastructures are becoming increasingly service-oriented. Information processing functionality is provided through interfaces (called service descriptions) that hide the underlying implementation. Service consumers do not depend on the actual binding of service implementation to description such that implementations can be changed or replaced without effect for the consumers. The collection of standards, methodological principles and technological frameworks that enable this highly flexible architectural view on IT infrastructures is known as Service-Oriented Architecture (SOA).

However this flexibility on the IT side is not always paired with flexibility on the business side (i.e. agility). Currently, Information Management research is investigating how service-orientation can also be applied to the business infrastructure of an organization [1]. A service-oriented business architecture would describe the individual steps to be executed within the organization's business processes as services delivered to these processes. These business services can then be matched with IT (or other) services that deliver the required functionality and business processes can be reconfigured by changing their mapping onto business services.

Foundational concepts underlying the idea of Service-Oriented Business Architecture (SOBA) have been pioneered by IBM's Research Division in its desire to lift service thinking from the IT to the business domain [2]. A motivating factor for introducing a distinctive Service Science research discipline was IBM's own successful transformation of hardware manufacturer into service business. Service

G. Pallis et al. (Eds.): ICSOC 2011, LNCS 7221, pp. 220–225, 2012.

Science has grown into an interdisciplinary academic initiative called Service Science, Management and Engineering (SSME) [3], which studies the service systems resulting from the 'servitisation' of business and society in general.

2 Research Objectives and Questions

The goal of the research project is to design a method for developing business architectures that are service-oriented. SOBA is a new concept that still has to be explored in-depth and further defined. Therefore our research is problem-centered [4]. The first two (groups of) research questions (RQ) aim at clarifying and analyzing the principles and characteristics of SOBA. The answer to these questions will provide a list of design criteria for the intended SOBA method, whose development and testing is the subject of the third RQ.

RQ1. The idea of SOBA originates in SOA and thus is an IT perspective on business architecture where architectural principles that work well for the IT domain are transferred to the business domain. What remains to be investigated is what service-orientation for business really means and how this service-orientation is or should be reflected in the way a business organizes itself (i.e., its business architecture). Foundational concepts of service-orientation are getting shape in the new discipline of SSME. RQs addressed by our research are: To what extent are these foundational concepts based on various business/management conceptual frameworks and theories of service? Can the relationships between these foundational concepts be uncovered by analyzing their mapping onto the concepts from these service theories? Can the concepts be integrated in a service system conceptualization or ontology?

RQ2. The result of RQ1 is a theory-based conceptual model that needs to be applied to architectural ideas for business. To this end, we need to get a better view on what service-orientation means in the context of business architectures. Starting from current SOBA literature and the developed service system model, we will investigate real business architectures and assess their service-orientation. RQs in this group are: Do companies that communicate to be service-oriented extend this service-orientation to the processes and structures within the company? Is service-orientation on the outside (i.e., towards the customer) always paired with service-orientation on the inside (i.e., the internal organization of the business)? How can service-orientation at the inside be characterized? Can a degree of service-orientation be defined and measured? Are there opportunities to improve the degree of service-orientation?

RQ3. RQ1 leads to a number of theoretical insights about the foundational concepts of a service system. RQ2 will provide us with empirical insights on how to model business architectures as service systems. It also allows assessing the difference between real-life practice and research on SOBA. This brings us to a third group of research questions: Can the knowledge of the problem domain gained from RQ2 be formalized into a list of design criteria for a service-oriented business architecture development method? Can such a method be designed? Will such a method improve the degree of service-orientation of business architectures?

3 Methodology

To achieve the research objectives, a Design Science approach will be taken, which is an accepted research methodology in the field of Information Management (see e.g. [5]). The primary Design Science artifact intended by our research is a method. To achieve this objective, a proper scientific approach will be taken based on the methodological guidance provided by Peffers et.al. [4]. Figure 1 shows which steps should be conducted in this Design Science research.

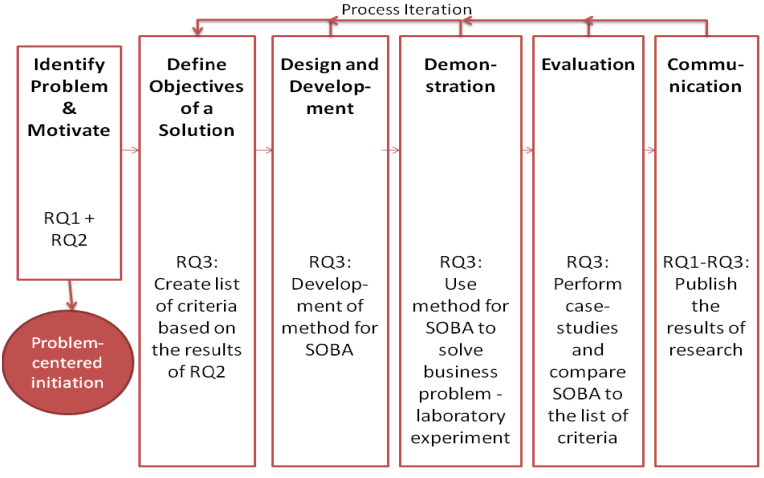


Fig. 1. Design Science Research Model: Process model (Source: [4])

First, the research problem should be identified and motivated. The research required for RQ1 has been completed and is discussed in section 6. The approach to solve RQ2 is twofold. On the one hand, a thorough literature study will be performed. The goal is to examine if and how service-orientation on the inside of companies can be realized as described in literature. Therefore we will analyse both EA literature as service management literature. On the other hand, we will execute interviews with domain experts in companies. For these interviews we will compose a questionnaire based on problems, challenges, and other findings of the literature study. The interviews will provide us with primary data that will give insight in the state of the practice of implementing SOBA.

Second, the objectives of the solution should be defined in RQ3. Therefore we will compose a list of design criteria based on the results of RQ2. These criteria will help to design and evaluate an EA method for business architecture development that can be used for implementing SOBA.

Third, to further tackle RQ3, we can start from an existing EA method, preferably a method with sufficient modeling support – at this moment TOGAF in combination with the ArchiMate EA modeling language seems to be the most likely candidate. Usually EA methods are used for the joint development of an application, IT and

business architecture. The part of the method that focuses on business architecture development will first be evaluated against our design criteria. Next we try to improve the service-orientation of business architecture development by mapping the conceptual model of RQ1 onto the constructs used by the EA method for business architecture modeling and changing or enhancing the EA method accordingly such that business architectures can be modeled as service systems. These changes should provide a better answer to the needs, problems and challenges defined based on the literature study and expert interviews from RQ2. A final step in the design process is to define the relationships between the concepts, structures and laws of the new (or transformed) business architectural model and those of the other architectural models prescribed by the EA method. The aim of this step would be to ensure that the changed/enhanced business architectural model still integrates with the application and IT architectural models, in particular when these are based on SOA principles.

Fourth, demonstration and evaluation of the new method will take place in two phases. First a laboratory experiment will be set up with a group of Information Management students. This experiment will demonstrate how the method can be used and will provide us with some feedback which can be used to evaluate and improve the method. The improved method will next be used in case-studies in companies, preferably companies that participated in the expert interviews. The case-study findings will hopefully prove the advantages of the developed method in terms of improved service-orientation of business architectures (or service-orientation at the inside) and resulting effects on service-orientation at the outside. We aim at a qualitative assessment of the new method.

4 Research Plan

Work Package	Research Activity	X = 01/09/2010
1.1	RQ1: Literature study of SSME and business/management theories of service. Development of conceptual model of service system	X + 0 – X + 8
1.2	RQ2: Literature study on SOBA and EA. Preparing conduct of field studies	X + 8 – X + 14
1.3	RQ2: Field studies - Observation of (service-oriented) business architecture in practice	X +14 – X + 18
2.1	RQ3: Develop design criteria for SOBA method	X + 18 – X + 20
2.2	RQ3: Design, development and demonstration (pilot testing) of SOBA method	X + 20 – X + 32
3.1	RQ3: Evaluation of SOBA method (case studies and/or laboratory experiments)	X + 36 – X + 42
3.2	RQ1-3: Communication of research results and finalize/defend PhD dissertation	X +42 – X + 48

5 Innovative Aspects and Scientific Contributions

The first scientific contribution is one to the new SSME discipline. The emerging theories that account for 'servitisation' and explain the dynamics of service systems have not yet been formalized in ontologies that can be used for modeling and simulating service systems [6]. The first work package in which RQ1 is analyzed can be viewed as a first step towards a service system ontology for service science. Furthermore, the results will also be used as a theoretical foundation for the design of a method for the development of service-oriented business architectures.

The second contribution is the additional knowledge of EA and more specifically of SOBA. The results of this PhD research will originate both from theory as from practice. First, an elaborate literature study on the service-orientation of business architectures and the confrontation of the results of this study with actual practice (RQ2) will enable researchers to better understand business as service systems. Second, the results of RQ3 will help companies to ensure that their outward service-oriented focus is mirrored in the activities and processes within the company. The research therefore contributes to the EA domain as well as to SSME (calling specifically for modeling and simulation tools for service systems [7]).

6 Preliminary Results

The results of RQ1 were summarized in a first conference paper which is accepted at the ICSOC 2011 conference.[1] In this paper we clarify the service systems worldview proposed by Service Science researchers Spohrer and Kwan [8] by investigating its foundational concepts from the perspective of established service theories and frameworks. By mapping the proposed service system concepts on the selected service theories and frameworks, we investigated their theoretical foundations, examined their proposed definitions and possible conflicting interpretations, discovered their likely relationships and general structure, and identified a number of issues that need further discussion and elaboration. This conceptual analysis resulted in a conceptual model which is shown in figure 2.

Our research points out that more or less all of the foundational concepts and their proposed specialisations are covered by one, many or in some cases even all reviewed service theories or frameworks. We identified a couple of issues that need further discussion and elaboration, e.g., because of conflicting views when mapping foundational concepts to the concepts of different service theories. Overall, however, our analysis shows that there is evidence of theoretical support for the proposed service systems worldview. All service theories and frameworks provide information on how the different service concepts are related to each other. Based on this knowledge we have drawn the relationships in the conceptual model represented in figure 2.

[1] Elisah Lemey and Geert Poels: Towards a Service System Ontology for Service Science. Full paper accepted for the 9th International Conference on Service Oriented Computing (ICSOC 2011), Paphos, Cyprus, 5-8 December 2011.

The conceptual model developed in this paper can be used as a basis for the further formalisation of the service systems worldview into a service systems ontology. The availability of a consensually agreed ontology could take Service Science a big step forwards as the integrative nature of the research intended by this interdisciplinary field requires a common ground to succeed.

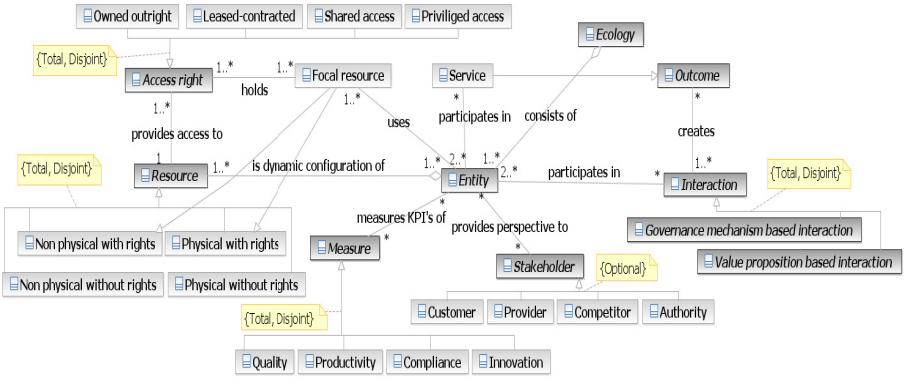


Fig. 2. Conceptual model of service systems worldview (UML Class diagram)

References

1. Cherbakov, L., Galambos, G., Harishankar, R., Kalyana, S., Rackham, G.: Impact of service orientation at the business level. IBM Systems Journal 44, 653–668 (2005)
2. Walker, L.: IBM business transformation enabled by service-oriented architecture. IBM Systems Journal 46, 651–667 (2007)
3. Spohrer, J., Maglio, P.P., Bailey, J., Gruhl, D.: Steps toward a science of service systems. Computer 40, 71–77 (2007)
4. Peffers, K., Tuunanen, T., Rothenberger, M.A., Chatterjee, S.: A design science research methodology for information systems research. Journal of Management Information Systems 24, 45–77 (2007)
5. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information systems research. Mis Quarterly 28, 75–105 (2004)
6. Wild, P.J.: A systemic framework for supporting cross-disciplinary efforts in services research. CIRP Journal of Manufacturing Science and Technology (2010)
7. IfM, IBM: Succeeding through service innovation: a service perspective for education, research, business and government. University of Cambridge Institute for Manufacturing, Cambridge (2008)
8. Spohrer, J., Kwan, S.K.: Service Science, Management, Engineering, and Design (SSMED): An Emerging Discipline-Outline & References. International Journal of Information Systems in the Service Sector (IJISSS) 1, 1–31 (2009)

Managing Things in an Ambient Space

Sujith Samuel Mathew[1]
Supervised by: Yacine Atif[2], Quan Z. Sheng[1], and Zakaria Maamar[3]

[1] School of Computer Science, University of Adelaide, Adelaide, Australia
[2] Faculty of IT, UAE University, Al Ain, UAE
[3] Zayed University, Dubai, UAE
{sujith,quanzheng.sheng}@adelaide.edu.au,
yacine.atif@uaeu.ac.ae, Zakaria.Maamar@zu.ac.ae

Abstract. We are surrounded by inanimate things that have inherent information. Unfortunately, the vast majority of applications built to use this information are built in ad-hoc manner, introducing issues with maintainability, share-ability and reusability. We discuss the architecture of Ambient Space Manager (ASM), a system to explore and control things within a context-aware ambient space. We define the context variables of such things to be the Capability, Location, Operations and QoS. Here we also elaborate the Capability context based on the atomic capabilities of things that enable them to offer contextual services.

Keywords: Web of Things, Context-aware Ambient Space, Future Internet.

1 Introduction

"*By 2015, wirelessly networked sensors in everything we own will form a new Web. But it will only be of value if the terabyte torrent of data it generates can be collected, analyzed and interpreted*" [1]. With billions of things around us, people find themselves in ambient environments (i.e., environments that provide seamless communication between people and things). Ambient applications make the management of large infrastructures like modern university campuses and multinational organizations easier to unleash the potential of their resources. Unfortunately, the vast majority of these applications are built in ad-hoc manners, introducing issues with maintainability, share-ability and reusability. Advances in Web services, Cloud computing, wireless networks and identification technologies, make processing power and communication capabilities available in increasingly smaller packages. Indeed, the Internet is evolving into the so-called "Web of Things" (WoT) or "Cloud of Things", an environment where everyday objects such as buildings, sidewalks, traffic lights, and commodities are identifiable, readable, recognizable, addressable, and even controllable via the Web [2,3,4].

With a plethora of things becoming ubiquitous on the Internet, there is a need to model scenarios for the WoT, and a plan to handle them in ambient environments. There are quite a few challenges in building an integrated system of heterogeneous

G. Pallis et al. (Eds.): ICSOC 2011, LNCS 7221, pp. 226–232, 2012.

physical things on the Web. Though the Semantic Web could meaningfully represent resources of physical things, there is a lack of standard interfaces to access these things by Web applications which use thing's capabilities in a given context. In addition, the heterogeneity of ubiquitous computing systems poses a major problem for system architects with respect to many protocols, component architecture, and data formats. There are no clear specifications of common characteristics of thing's attributes and processes, for controlling them or querying them over the Web. An architecture definition is required for managing things and we propose a system architecture, to manage the access and control of things in an ambient space.

2 Related Works

A contributing factor to the fast and widespread growth of the Internet is the increasing dependence on the Internet as an economical and efficient means of communication. The increasing availability of Internet access points and enhanced infrastructures of whole cities to support wired and wireless Internet connection are fueling this trend [5]. Bodies such as the IP for Smart Objects alliance (IPSO) [6] and the European Future Internet Initiative (EFII) [7] have also accelerated this trend to connect a variety of physical things into the Internet, with the intention of propagating and managing the wide use of Internet as the common medium for communication. Guinard and Trifa [8] successfully demonstrated Web mashups by exposing real world things as RESTful Web services. Their research compares two ways of interfacing real-world devices into the Web by having Web servers embedded in devices and secondly by connecting devices to an external proxy Web server, as a gateway. A recent product launched by OpenPicus, the FlyPort [9] hosts a compact Wi-Fi module which is 35X488 mm. The FlyPort supports Web server and socket applications. With digital inputs and output ports, this miniature Wi-Fi module can be augmented to any device to Web enable them. Michael Beigl et al. [10] define smart physical things as things augmented with computing and communication capabilities, which can be accessed by computer applications. Similarly, Friedemann [11] envisions smart things to be able to wirelessly communicate with people and other smart things, with the ability to understand the presence of surrounding objects. Today, these definitions do not formally encompass all things that could be on the Web, e.g., an RFID tagged chair or a Personal Digital Assistant (PDA), both can be accessed on the Internet.

3 System Architecture for Managing an Ambient Space

We propose the architecture of Ambient Space Manager (ASM), designed to manage the access and control of things in an ambient space. ASM provides a gateway to things on the Web for building ubiquitous applications. This architecture creates a support system for ambient spaces where ubiquitous things are seamlessly accessed.

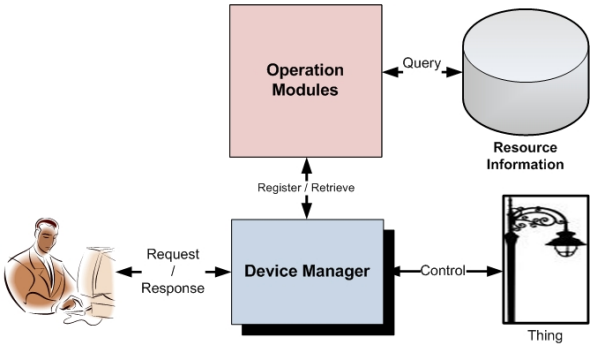


Fig. 1. Modules of the Ambient Space Manager (ASM)

To establish a separation of concerns in dealing with heterogeneous things in an ambient space, a layered architectural approach is required to provide modularity and integration patterns based on which ambient client applications are developed. As shown in Figure 1, the Device Manager interfaces users/agents with physical things in the ambient space whereas Operation Modules register and retrieve relevant information to facilitate the interaction with physical things. All relevant things within an ambient space are registered with ASM. Further, the Resource Information module encapsulates the knowledge base of things and related context information.

3.1 Web-Enabling Things

A thing becomes Internet-enabled if it is associated with networking capability (i.e. has an IP address), which uniquely identifies it within the Ambient Space (Figure 2a). A thing becomes Web-enabled when it is augmented with a Web server (Figure 2b) so that it can expose its functional and non-functional capabilities on the Web through HTTP. Though arguably, there is scope for WS* and REST in the area of Web services, advances in REST based Web service architectures is propagating the abstraction of physical things as services on the Web. This trend gives rise to the

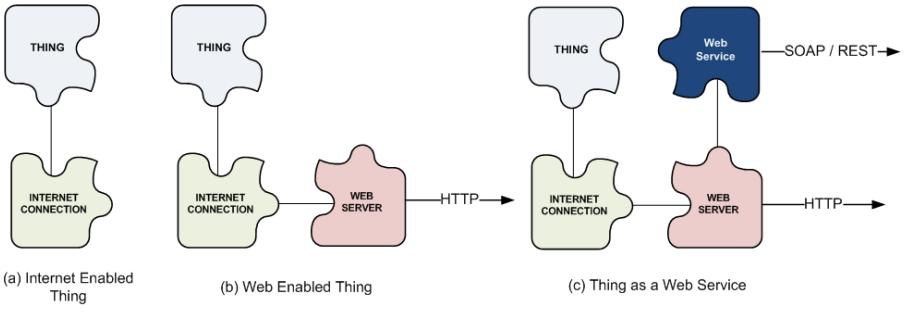


Fig. 2. Connecting things to the Web

possibilities of wrapping things in the physical world as Web services (Figure 2c). The FlyPort module [10] is a Wi-Fi module, hosting a tiny Web server and multiple power output points. This configuration enables it to be networked and controlled on the Web. The output points are connected to a thing, say a light as shown in Figure 3 and then controlled from the Web. The Web-enabled Smart light is registered with ASM, redefining its context within the ambient space.

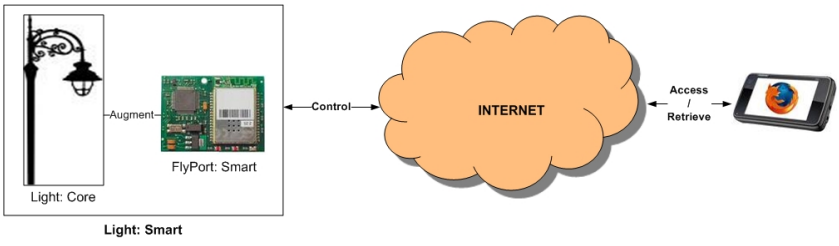


Fig. 3. Light augmented to be accessed on the Web

3.2 Context of Things in an Ambient Space

To address the challenge of managing things in an Ambient space, we define the context of things within an ambient space by its Capabilities, Location, Operations, and Quality of Service (QoS) labeled thereof as CLOQ. An initial configuration of ASM requires CLOQ context to be registered for all things within an ambient space. To represent the commonalities of things, a formal ontology is a necessity [3] and hence we also elaborate the ontological structure based on the capabilities of things. As presented in our earlier work [12], the Capability dimension is a formal ontology of thing's capabilities to support the design of systems in ambient spaces. This ontology proposal is specific to the application context of the proposed ASM. Further extension of this work aims at developing a bridging scheme to share common things specifications on WoT.

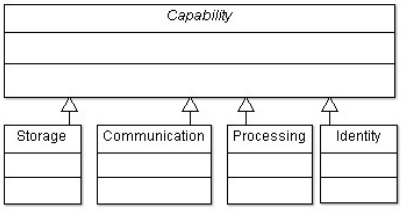


Fig. 4. UML: The abstract Capability class and IPCS sub-classes

Things on the Web are classified using four fundamental, atomic dimensions that characterize their intrinsic capabilities i.e. Identity, Processing, Communication and Storage referred to as the IPCS set, as shown in Figure 4. ***Identity (I)***: A thing must be uniquely identifiable with the use of an identification system. A thing could adopt

multiple identification systems (e.g., Bluetooth address and IP address). Identity is the mandatory and minimal requirement for things to be managed by ASM. ***Processing (P)***: The processing capability of a thing is a system that has functions which allow a thing to be controlled or managed. ***Communication (C)***: The communication interface is a system enabling interaction with the thing describing the input, output, or both. ***Storage (S)***: Storage is a system that describes the information that a thing retains abstracting the states and values.

Based on the IPCS capabilities, we define the taxonomy of things on the Web. A ***Core*** thing has the bare capability of being uniquely identified within a given context. A ***Primitive*** thing has a unique identity and includes one additional capability of the other three dimensions. These are further categorized as the Fat, Plug and Fuzzy. A ***Complex*** thing has a unique identity and combines two of the other three dimensions. A ***Smart*** thing combines all the capabilities of the IPCS set. The combinations are shown in Table 1 where, *Identity* is not included as it is mandatory for all categories.

Table 1. Classification of Primitive Things and Complex Things

Primitive Things				
Names	P	C	S	Examples
Fuzzy	X			Washing machines, Microwave ovens, etc.
Plug		X		Headphones, Speakers, etc.
Fat			X	CD, DVD, etc.

Complex Things				
Names	P	C	S	Examples
Social	X	X		Remote controls, land-line phones, etc.
Sticky		X	X	USB stick, RFID tag, etc.
Gizmo	X		X	Calculator, Game Boy, etc.

3.3 Implementation Framework

Interaction with ASM is through the Device Manager (Figure 1). The user searches from things registered with ASM, based on CLOQ context. The Device Manager sends requests to the Operation modules which initiate the discovery of the requested thing. The search is based on the capabilities of things in the knowledge base within the Resource module. The knowledge base is an ontological classification of things based on IPCS capabilities. Querying the ontology is done using W3C recommended SPARQL language for RDF. The search potential of SPARQL is appropriate for querying [14] the knowledge base. The logic for dynamically querying the knowledge base is implemented using Jena APIs [13], an open source framework for building Semantic Web applications. Jena is Java based and provides an environment to programmatically access the ontology. The knowledge base is queried with Jena APIs, like `executeSPARQLQuery()`, that takes a SPARQL query as a parameter and runs it against the knowledge base (example shown below).

```
JenaOWLModel owlModel =
             ProtegeOWL.createJenaOWLModelFromReader(input);
String queryStr = "SELECT ?primary
             WHERE ?primary rdfs:subClassOf WOT:Primary";
QueryResults results =
             owlModel.executeSPARQLQuery(queryStr);
```

The query is parsed and the result is processed by the Operation Layer. From the list of things with the requested capabilities, their Location, Operations and QoS are filtered based on the request. The information is then made available, through the Device Manager over the Web to the external system.

4 Conclusions

The proposed architecture provides a modular approach to manage context-related information of things on the Web. The proposed contributions help to realize ambient spaces where real-world things seamlessly interact with each other to provide new services. We continue our work in further refining the context variables of Capability, Location, Operations and QoS. These variables form the building blocks of ASM to encapsulate the functions of everyday things as services on the Web. Hence a comparative study of existing Web service solutions is forthcoming. Considering the fact that things can be dynamic in space and time, managing the presence of things, composition of things and the significance of security and privacy is an interesting area of study for the future.

References

1. Raskino, M., Fenn, J., Linden, A.: Gartner, http://www.gartner.com/DisplayDocument?id=476440 (retrieved on May 14, 2011)
2. Mulligan, G.: The internet of things: Here now and coming soon. IEEE Internet Computing 14(1), 35–36 (2010)
3. Raggett, D.: The Web of Things: Extending the Web into the Real World. In: van Leeuwen, J., Muscholl, A., Peleg, D., Pokorný, J., Rumpe, B. (eds.) SOFSEM 2010. LNCS, vol. 5901, pp. 96–107. Springer, Heidelberg (2010)
4. Berners-Lee, T.: The web of things. special theme on the future web. ERCIM News – the European Research Consortium for Informatics and Mathematics (2008), http://ercim-news.ercim.eu/en72/keynote (retrieved on March 12, 2011)
5. Danigelis, A.: 10 cities with widespread wireless, http://dsc.discovery.com/technology/tech-10/wireless-cities-top.html (retrieved on March 12, 2011)
6. IPSO Alliance. Promoting the use of ip in networks of smart objects, http://www.ipso-alliance.org/Documents/IPSO_Briefing.pdf (retrieved on March 13, 2011)

7. The European Future Internet Initiative. White paper on the future internet PPP definition, http://www.futureinternet.eu/fileadmin/documents/reports/Future_Internet_2020-Visionary_Panel.pdf (retrieved on March 12, 2011)
8. Guinard, D., Trifa, V.: Towards the web of things:Web mashups for embedded devices. In: MEM 2009 in Proceedings of WWW 2009. ACM (2009)
9. Openpicus, flyport module, http://www.openpicus.com/cms/ (retrieved on March 12, 2011)
10. Beigl, M., Gellersen, H.W., Schmidt, A.: Mediacups: experience with design and use of computer-augmented everyday artifacts. Computer Networks 35(4), 401–409 (2001)
11. Mattern, F.: From smart devices to smart everyday objects. In: Proceedings of Smart Objects Conference (January 2003)
12. Mathew, S.S., Atif, Y., Sheng, Q.Z., Maamar, Z.: Ambient things on the web. Journal of Ubiquitous Systems and Pervasive Networks (JUSPN) 1(1), 1–8 (2010)
13. Jena - a semantic web framework for java, http://jena.sourceforge.net/ (retrieved on May 14, 2011)
14. Sirin, E., Parsia, B.: SPARQL-DL: SPARQL query for OWL-DL. In: 3rd OWL Experiences and Directions Workshop (2007)

A Propagation Model for Integrating Web of Things and Social Networks

Lina Yao
Supervised by: Michael Sheng

School of Computer Science
The University of Adelaide, Adelaide SA 5005, Australia
lina.yao@adelaide.edu.au

Abstract. Modeling interest of a user for services recommendation and friendship between users is the major activity of social networks. The information used by social networks such as user profiles is unfortunately easy to be faked and misled by the users, which often results in poor service recommendation and friendship prediction. In this paper, we propose a propagation model that integrates the emerging Web of Things (resource/services networks) and social networks together so that better service recommendation and friendship prediction can be achieved by considering interactions between people and things.

1 Introduction

Over the years, the Web has gone through many transformations, from traditional linking and sharing of computers and documents, to current connecting of people via numerous social networks such as Facebook and Linkedin. The main goals of social networks are to effectively model the interest of a user and the friendship between users, which support a number of useful applications such as service recommendation based on user's interest and friendship prediction [14]. With the recent advances in radio-frequency identification (RFID) technology and Web services [11], the Web will continue the transformation and will be slowly evolving into the so-called "Web of Things and Services". This future Web will provide an environment where everyday physical objects such as buildings, sidewalks, and commodities are readable, recognizable, addressable, and even controllable using services via the Web.

Existing research on social networks are mainly based on mining i) the users' profiles, which are the meta-data from users when they register on the social networks, ii) comments and topics posted by the users, and iii) the interactions between users (e.g., news the users share and games the users play). All these sources are unfortunately easy to be faked and misled by the users. This often results in poor service recommendation and friendship prediction. With the emerging Web of Things, a new additional dimension becomes possible to improve service recommendation and friendship prediction by studying the relationship between things and people. Based on the social phenomenon of homophily [5], people who access similar resources tend to have similar interest and are more likely to be friends.

In this paper, we propose to connect Web of things and services with Web of people (i.e., social networks) together. This connection enables better service recommendation

G. Pallis et al. (Eds.): ICSOC 2011, LNCS 7221, pp. 233–238, 2012.

and friendship and human behavior prediction based on analyzing interactions between people and things. Two major techniques are widely used in social networks analysis. Content-based recommender systems make recommendations by analyzing the content of textual information and finding regularities in the content, while collaborative filtering (CF) techniques use user-item ratings data to make predictions and recommendations [12]. Both approaches have limitations. CF systems do not explicitly incorporate feature information and content-based systems do not necessarily incorporate the information in preference similarity across individuals. Hybrid CF techniques (e.g., the content-boosted CF algorithm [6] and Personality Diagnosis (PD) [9]), which combine CF and content-based techniques, hold the hope to avoid the limitations and thereby improve recommendation performance. In this paper, we propose to use a hybrid methodology that combines collaborative filtering techniques with knowledge-based recommending techniques, to model the interactions between people and things.

This paper discusses the research problem and significance (Section 2), our proposed model to study three relationships among people and things, as well as some future work ahead (Section 3).

2 Problem Description and Research Significance

In a Web of Things environment, physical things connect to the Internet offering resources and services. Social networks are designed to connect people over the Internet where people can interact with each other, share resources and create their own community. We propose to establish an integrated network that links social networks and Web of things together. In such a network, users not only find their interested people, but also their interested services and resources provided by the things. Users' interest and friendship based on social networks can propagate to the Web of Things.

There are two kinds of entities considered in this work: *people* and *things*, which generate three kinds of interactions, namely *people to people*, *people to things*, and *things to things*. Our goal is to propose an integrated framework that can better predict and recommend the most appropriate services and things to users. In particular, we will analyze two behaviours: *user behaviour* and *thing behaviour*. User behaviour can be learned from the people to people network (e.g., friendships in social networks) and people to things network (e.g., interactions between users and things), while thing behaviour can be extracted from the people to things network and things to things network (i.e., linkage among things).

Unfortunately, the information needed to deduce user or thing behaviors are usually noisy, in large volume, and even missing. Inferring missing elements (also called latent variables) from observable elements is a challenging task. The key task in our work is to associate the latent variables with both users and things, to define coupled models that encode relationships between people and things. In particular, we will define a shared latent variable to assure dynamical and evolutionary interaction between Web of things and social networks during the learning process.

2.1 Research Significance

Existing researches in social networks focus on analyzing users' interests by profiling their information from the users' profile meta-data or inducing users' interests by digging their behaviors among the social networks. Recently, researchers have begun to consider the effects from user-user friendships, which have the impact on services recommendation. For example, user-user friendships are taken into account to recommend the most appropriate services, e.g., advertisements and news items [14]. However, very few work involve things in the services recommendation applications.

Existing recommendation systems work well in the traditional services-user environment, like collaborative filtering (CF) techniques. However, the precondition of such systems is that users are independent. In our model, users' interactions are an important factor and they are correlated other than propositional entities, i.e., the users' interactions can affect the decision making process of the recommendation results. In addition, things can be composed to complete a certain service. Things can come and go, and are very uncertain in terms of service offering. This will result in well known *cold-start* problem in recommendation systems. Cold-start problems occur when a new person or new thing comes into the network, there is no or little initial information available for them [13].

The key idea of our work is to associate the latent variables with both people and things and encode information about the people to people friendship, people to thing interactions and thing to thing linkage. With our result, it is possible to further perform mining—based on the interactions and relationship between people and things—to predict human activities and behaviors. The relationships can be understood as (i) behavior of user by using things, (ii) interactions between users and things (e.g., where the user shows up and her location information), and (iii) correlations among things.

3 Proposed Solution

The nontrivial correlation between Web of things and social networks motivates joint modeling of both sources of evidence. The thing to people propagation model simultaneously encodes the three heterogeneous types of dyadic relationships, namely *people to things* ($\{\exists y_{i,j}, \forall i \in \mathcal{P}, j \in \mathcal{S}\}$), *people to people* ($\{\exists r_{i,i'}, \forall i, i' \in \mathcal{P}\}$), and *things to things* ($\{\exists t_{i,i'}, \forall i, i' \in \mathcal{S}\}$), where $\mathcal{P}$ and $\mathcal{S}$ denote people and things respectively. We consider all three relationships as directed graph, and model them based on the latent variable graphical model.

3.1 Modeling People to Things

This can be considered as a recommendation system and its aim is to offer a user p_i the most appropriate things/services s_j. We propose to use the collaborative filtering (CF) techniques in the modeling, which is based on learning the past interactions between people and things. In particular, neighborhood-based collaborative filtering is one of prevalent memory-based CF that assumes the interactions between user p_i and a thing s_j depends on the observations of neighbouring users or things [10]. Similar things or services can be propagated to a particular user [14].

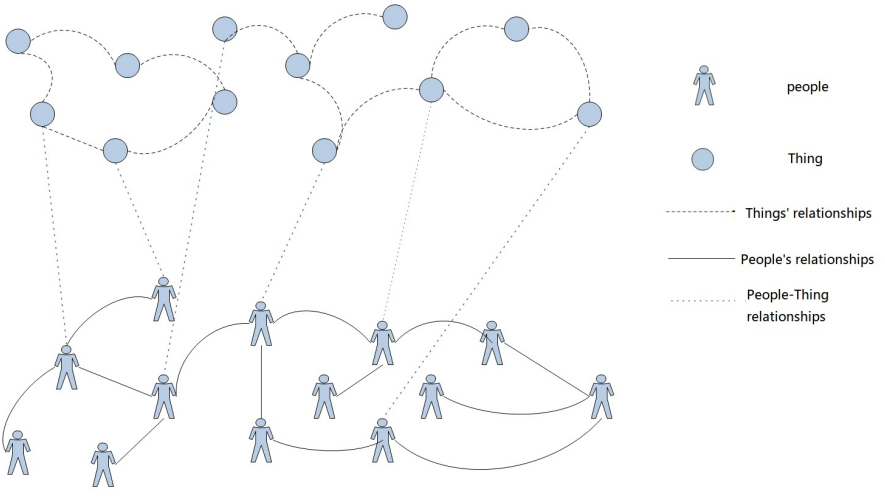


Fig. 1. Integrating Web of things and social networks: all the connections include unipartite edges within the people to people network, bipartite user to things and unipartite things to things

We assume that for each person p_i and each thing or service s_j, their observable variables are x_i, which includes the person's registration information and profile etc, and x_j, which is the thing's description respectively. Moreover, the hidden variable z_i refers to the person's interests that are not observed directly and z_j refers to the thing's semantic information or topics. So $p(y_{i,j})$ depends on the latent and observable variables which is denoted by $p(y_{i,j}|(i,j) \sim p(y_{i,j}|x_i, x_j, z_i, z_j; \Theta)$, where Θ is a collection of hyper parameters. It should be noted that we must consider the cold-start problem since new things keep adding easily to the Internet while old things may disappear. We will refer to [1] to deal with the cold-start problem.

We define x_i and x_j are the observable variables for the people and things based on its defined dependency, their relationship will be: $z_i \sim p(z_i|x_i)$, $z_j \sim p(z_j|x_j)$, and z_i and z_j use their inner product $z_i^T z_j$. By coupling the neighbourhood dependency with the latent variable model together based on [2,3], we have:

$$\hat{z}_i = \frac{\sum_{i' \in \Gamma_i} w_{i,i'} z_{i'}}{\sum_{i' \in \Gamma_i} w_{i,i'}}$$

and $p(y_{i,j}|(i,j)) \sim p(y_{i,j}|x_i, x_j, \widehat{z}_i z_j; \Theta)$, where $w_{i,i'}$ is measure the similarity between people i and its neighbour $i' \in \Gamma_i$.

3.2 Modeling People to People

The people to people relationship is modeled using the Random Walk on graph. A random walk is a finite Markov chain that is time reversible [4], which starts at a node (each person can be considered as a node) i and iteratively moves to a neighbor of i chosen uniformly at random. The hitting time $h_{i,i'}$ from i to i' is the expected number

of steps required for a random walk starting at i to reach i'. In our model, we define the graph to be a undirected graph, so that the commute time $c_{i,i'} := h_{i,i'} + h_{i',i}$. We normalize the hitting time, so these measures can be considered as proximity (similarity) measure $r_{i,i'} := h_{i,i'} \cdot \pi_{i'}$ for each pair of people (i, i'). Moreover, in our model, the transition probability $p(i, i') = r_{i,i'}/d_i$ and $p(j, j') = t_{j,j'}/d_j$, where d_i and d_j are the degree of i and j respectively. Based on the same assumption [14], assuming user i is fully characterized by observable features x_i and latent variable z_i, we have $r_{i,i'} \sim p(r_{i,i'}|x_i, x_{i'}, z_i, z_{i'}; \Theta)$.

3.3 Modeling Things to Things

We consider the things to things relationship as a graph. The similarity between things is measured based on the non-functionalities of things. We construct the similarity graphs using Preferential attachment to model thing to thing relationship proximity. Its basic idea is the probability that a new connection involves node (thing) j is proportional to $|\Gamma(j)|$, the current number of neighbors of j [7], so the $s_{j,j'} := |\Gamma(j)| \cdot |\Gamma(j')|$ [8].

For the things s_j and s'_j, we connect s_j and s'_j if s'_j is among the k-nearest neighbors of s_j. We further measure their pairwise similarities $t_{j,j'} = t\{s_j, s'_j\}$ by the similarity function we defined which is symmetric and non-negative, and so that the corresponding similarity matrix $\mathcal{Q} = t_{j,j'}$, where $j, j' \in \{1, n\}$. We also consider each thing depends on its observable factors x_j and latent factors z_j. Finally, we have $t_{j.j'} \sim p(t_{j,j'}|x_j, x_{j'}, z_j, z_{j'}; \Theta)$.

Our model integrates the people-people relationship model $(i, i', r_{i,i'})$, people-thing relationship model $(i, j, y_{i,j})$ and thing-thing relationship model $(j, j', t_{j,j'})$ together and there are two tuples of sharing factors, namely (i, x_i, z_i) and (j, x_j, z_j). Our model should learn the shared parameters and latent variables from integrated relationships. In general, latent variable graphical models have a significant difficulty for model selection because one may not know the number of relevant latent variables, nor the relationship between these variables and the observed variables. Convex formulations require the manipulation of a full matrix which is impractical for anything beyond thousands of participants in our model. In addition, relationships between users and things change over time, which calls for a learning algorithm that are efficient and amendable to dynamic data streams.

We will build up a testbed to verify and test our latent graphical model, based on popular social networks such as Facebook and Linkedin, as well as things network connected by RFID and sensors. We will also conduct some comparison experiments to demonstrate that our model can produce more accurate and efficient recommendations.

4 Conclusion

The emerging Web of things and services provides an exciting opportunity to improve the poor performance of services recommendation and friendship prediction in today's social networks. This PhD work focuses on investigating ways to efficiently integrate social networks and Web of things together by considering people to people relationships, things to things relationships, as well as people to things relationships. In this

paper, we have presented some initial ideas of this work, as well as some preliminary design. Currently, we are constructing the integrated model. Our future work also includes verifying the proposed model using real social networks.

References

1. Agarwal, D., Chen, B.: Regression-based latent factor models. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 19–28. ACM (2009)
2. Koren, Y.: Factorization meets the neighborhood: A multifaceted collaborative filtering model. In: Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 426–434. ACM (2008)
3. Koren, Y.: Factor in the neighbors: Scalable and accurate collaborative filtering. ACM Transactions on Knowledge Discovery from Data (TKDD) 4(1), 1 (2010)
4. Lovász, L.: Random walks on graphs: A survey. Combinatorics, Paul Erdos is Eighty 2(1), 1–46 (1993)
5. McPherson, M., Smith-Lovin, L., Cook, J.M.: Birds of a feather: Homophily in social networks. Annual Review of Sociology 27(1), 415–444 (2001)
6. Melville, P., Mooney, R., Nagarajan, R.: Content-boosted collaborative filtering for improved recommendations. In: Proceedings of the National Conference on Artificial Intelligence, pp. 187–192. MIT Press (2002)
7. Mitzenmacher, M.: A brief history of generative models for power law and lognormal distributions. Internet Mathematics 1(2), 226–251 (2004)
8. Newman, M.: Clustering and preferential attachment in growing networks. Physical Review E 64(2), 025102 (2001)
9. Pavlov, D., Pennock, D.: A maximum entropy approach to collaborative filtering in dynamic, sparse, high-dimensional domains. In: Advances in Neural Information Processing Systems, pp. 1465–1472. MIT Press (2002)
10. Sarwar, B., Karypis, G., Konstan, J., Reidl, J.: Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th International Conference on World Wide Web (WWW 2010), pp. 285–295. ACM (2001)
11. Sheng, Q.Z., Zeadally, S., Luo, Z., Jung, J.-Y., Maamar, Z.: Ubiquitous rfid: Where are we? Information Systems Frontiers 12(5), 485–490 (2010)
12. Si, L., Jin, R.: Flexible mixture model for collaborative filtering. In: Proceedings of the 20th International Conference on Machine Learning (ICML 2003), vol. 20, p. 704 (2003)
13. Su, X., Khoshgoftaar, T.M.: A survey of collaborative filtering techniques. Advances in Artificial Intelligence, 4:2 (January 2009)
14. Yang, S., Long, B., Smola, A., Sadagopan, N., Zheng, Z., Zha, H.: Like like alike: Joint friendship and interest propagation in social networks. In: Proceedings of the 20th International Conference on World Wide Web, pp. 537–546. ACM (2011)

ERP B3: Business Continuity Service Level Agreement Translation and Optimisation*

Ulrich Winkler and Wasif Gilani

SAP Research,
The Concourse, Queen's Road, Belfast BT3 9DT, United Kingdom
{ulrich.winkler,wasif.gilani}@sap.com
http://www.sap.com/research

Abstract. ERP B3 is a SLA@SOI framework based solution for Service Level Agreements driven, on demand and dynamic provisioned business applications. We extended the ERP B3 framework with a process-centric Business Continuity Analysis toolkit. In this demo we want to show how our toolkit helps Business Continuity Manager (a) to estimate business impact of failed IT services (b) to validate SLAs and verify if selected SLAs are appropriated and (c) to determine an optimal set of service-level SLAs to minimise business disruptions.

Keywords: SLA, SLA@SOI, SLA SLA translation, SLA planning, Business Continuity Management, BCM.

1 Introduction

The SLA@SOI framework provides comprehensive support for holistic and transparent SLA management, SLA translation and SLA negotiation [1,2]. Lessons learned from using the SLA@SOI framework to implement ERP B3 solution to perform IT centric SLA translation are discussed in in [3].

We extended ERP B3 and added components which helps Business Continuity Manager (a) to estimate business impact of failed IT services (b) to translate business level requirements to IT-service and facility-level requirements and (c) to select an optimal set of service-level and facility-level SLAs [4]. The overall objective is to minimise business disruptions.

Our toolkit is very interactive and supports the Business Continuity Manager to explore potential alternatives and to answer "what-if-questions". The best way to explain these interactive modelling and optimisation components of our toolkit is with a demonstration.

* The research leading to these results is partially supported by the European Community's Seventh Framework Programme (FP7/2001-2013) under grant agreement no.216556.

G. Pallis et al. (Eds.): ICSOC 2011, LNCS 7221, pp. 239–240, 2012.

2 Outline of the ERP B3 BCM Demonstration

1. *BCM:* We will educate the audience about business process modelling, business performance analysis, and the role and importance of Business Continuity Management.
2. *Business Impact:* We extended a process modelling tool with business impact annotation modelling support. We will demonstrate how a business continuity manager uses this impact modelling support to model financial, legal and other business requirements in a process-centric way.
3. *Dependency and Risk Modelling:* We designed and implemented a dependency and risk modelling tool to model the IT service and facility hierachy. This tool enables the continuity manager to annotate risk to the service hierachy.
4. *SLA translation, selection and validation:* We explain how the continuity manager links the service hierarchy to business activities. Once this linkage has been established our tool translates business level requirements down to service level requirements and verifies if selected SLAs are suitable to fulfil business level requirements.
5. *SLA optimisation:* If more than one SLA has to be selected the SLA selection and validation problem becomes an optimisation problem. We explain how our tool solves this problem and proposes an optimal set of SLAs.

3 Conclusions

ERP B3 makes use of the SLA@SOI framework to provide service level aware on-demand business applications. In this demo we show how we extended the ERP B3 framework with Business Continuity Analysis support. We show how a continuity manager uses our extension to translate business level requirements down to service level requirements. An optimisation components helps the continuity manager to select a set of optimal SLAs.

References

1. SLA@SOI project: IST- 216556; Empowering the Service Economy with SLA-aware Infrastructures, http://www.sla-at-soi.eu/
2. Theilmann, W., Happe, J., Kotsokalis, C., Edmonds, A., Kearney. K.: A Reference Architecture for Multi-Level SLA Management. Journal of Internet Engineering (2010) (to appear)
3. Theilmann, W., Winkler, U., Happe, J., de Abril, I.M.: Managing On-Demand Business Applications with Hierarchical Service Level Agreements. In: Berre, A.J., Gómez-Pérez, A., Tutschku, K., Fensel, D. (eds.) FIS 2010. LNCS, vol. 6369, pp. 97–106. Springer, Heidelberg (2010)
4. Winkler, U., Gilani, W., Fritzsche, M., Marshall, A.: Model-Driven Framework for Process-centric Business Continuity Management. In: Seventh International Conference on the Quality of Information and Communications Technology, Porto, pp. 248–252 (2010)

Business Process Variability: A Tool for Declarative Template Design*

Pavel Bulanov, Heerko Groefsema, and Marco Aiello

Distributed Systems Group, Johann Bernoulli Institute, University of Groningen, Nijenborgh 9, 9747 AG Groningen, The Netherlands
{p.bulanov,h.groefsema,m.aiello}@rug.nl

Abstract. To lower both implementation time and cost, many Business Process Management tools use process templates to implement highly recurring processes. However, in order for such templates to be used, a process has to adhere substantially to the template. Therefore, current practice for processes which deviate more than marginally is to either manually implement them at high costs, or for the business to inflexibly comply to the template. In this paper, we describe a tool which demonstrates a variability based solution to process template definition.

Template utilization is a well known practice in the BPM domain. As an attempt to lower development time and costs, the practice is not without issues. One such an issue is that processes must adhere significantly to the template in order to achieve the goal of lowering implementation costs. We argue that the introduction of variability to BPM will provide solutions to this and other well known BPM issues [2]. When introduced to the BPM domain, variability indicates that parts of a business process remain either open to change, or not fully defined, in order to support different versions of the same process depending on the intended use or execution context [3]. In this paper, we briefly illustrate a modelling tool demonstrating a variability based solution to process template definition. The tool itself is based upon the Eclipse GMF framework and supports modelling of business processes using the BPMN notation. By extending the BPMN notation with the graphical elements of Process Variability: Declarative'n'Imperative (PVDI), the process modeller is provided with a large degree of template reusability and design flexibility. See [4] for a detailed description.

As an example, consider that the Netherlands consists of a total of 418 highly different municipalities which all have to implement the same services and laws created by the national government. These services and laws however may be implemented at the municipality's own discretion with regard to local requirements. One law which is highly subject to local needs is the WMO (Wet maatschappelijke ondersteuning, Social Support Act, 2006), a law which allows citizens with physical problems to be part of the community by providing wheelchairs, help at home, and home improvements. Figure 1 illustrates the tool used to implement and validate a variant of the WMO process found at a local municipality. Using domain specific information, in this case processes found at several municipalities,

* The research is supported by the NWO SaS-LeG project, http://www.sas-leg.net, contract No. 638.001.207.

G. Pallis et al. (Eds.): ICSOC 2011, LNCS 7221, pp. 241–242, 2012.

a process designer can implement a template. This template may range from being either a set of activities to a fully implemented business process. The process designer then introduces a number of PVDI constraints [4] to the template by including extra flows and groups in the business process model. In this case included were a frozen group (I), which allows no changes to its contents, a parallel link (H) and a number of constraint flows (A through G) which enforce elements or groups to follow each other either eventually (A,B,C,E,F,G) or immediately (D), and in a path (A,B,E,F,G) or all paths (C,D). From the finished template variants can then be created and validated. When creating variants a designer may adapt anything in the template except the PVDI elements. When ready, the variant can be validated regarding its compliance with the constraints defined within the template, and any element detecting problems within the variant will display an error (D,G). All the while, the complications of the underlying logic used for validation is hidden from the designer through the easy to understand graphical PVDI elements. The details of each of the elements used and their underlying validation logic can be found in [4]. Furthermore, a more detailed description of this entire life-cycle is available as a video at [1].

In this paper, we illustrated a tool which implements a new process template design technique offering high process design reusability and flexibility, while hiding underlying complexity via simple well understood design elements.

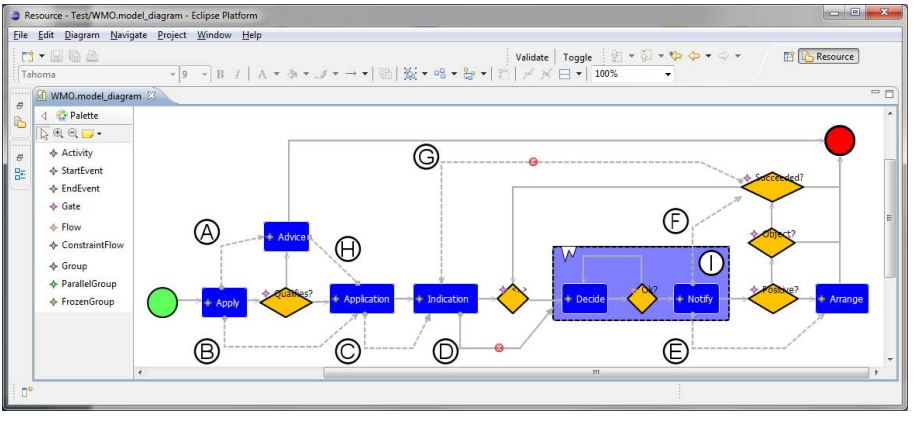


Fig. 1. Screenshot: Validating a variant

References

1. Demo video (2011), http://www.youtube.com/watch?v=coWkSSRpHD8
2. van der Aalst, W., Jablonski, S.: Dealing with workflow change: Identification of issues and solutions. International Journal of Computer Systems, Science, and Engineering 15(5), 267–276 (2000)
3. Aiello, M., Bulanov, P., Groefsema, H.: Requirements and tools for variability management. In: IEEE Workshop on Requirement Engineering for Services (REFS) at IEEE COMPSAC (2010)
4. Groefsema, H., Bulanov, P., Aiello, M.: Declarative Enhancement Framework for Business Processes. In: Kappel, G., Maamar, Z., Motahari-Nezhad, H.R. (eds.) ICSOC 2011. LNCS, vol. 7084, pp. 495–504. Springer, Heidelberg (2011)

A Cloud-Based Workflow Management Solution for Collaborative Analytics

Henry Kasim, Terence Hung, Xiaorong Li, William-Chandra Tjhi, Sifei Lu, and Long Wang

Institute of High Performance Computing,
Agency for Science, Technology and Research (A*STAR), Singapore
{kasimh,terence,lixr,tjhiwc,lus,wangl}@ihpc.a-star.edu.sg

Abstract. The concept of collaborative analytics is to accommodate reuse and collaboration in data analysis process through sharing of analytics methods, algorithms, and computation resources. However, realizing collaborative analytics is challenging due to the large data sets, high throughput and computational intensive requirements. In this demonstration, we present a cloud-based workflow management solution that allows collaborative analytics to run in the cloud computing environment. Our solution provides sharing of analytics resources, recommendation of analytic workflows, dynamic scheduling and provisioning for scalable data analytics, high availability through fault-tolerance, real-time monitoring and tracking of collaborative analytics status. Examples of a generic data mining analysis and climate change analytics are given to show that our work can be applied for a wide variety of study in the real-life world.

1 Introduction

Unraveling useful insights from raw data in a complex domain usually requires interconnecting data collection, preprocessing, analysis and post-processing steps into a sophisticated analytics workflow. In the real world, this workflow design is often done inefficiently through a dynamic and continuous process. To improve this process, analysts today resort to systematic sharing and reuse of analytics methods and resources, resulting in collaborative analytics.

Through collaborative analytics, user is assisted in workflow design as a wide-range of shared analytics workflows designed on collaborative environment are made available for reuse. With collection of workflows to consider, there is also a challenge to run multiple workflows concurrently without have to concern about the limitation of the computing resources. Due to the above challenges, we present a cloud-based workflow management solution for collaborative analytics, as shown in Fig 1.

In Fig 1, Step 1, a variety of data analytics workflows can be saved and shared to *Collaborative Analytics Workflow Database*. Step 2, a user submits his/her data into the collaborative analytics platform and requests for collaborative analytics solutions.

G. Pallis et al. (Eds.): ICSOC 2011, LNCS 7221, pp. 243–245, 2012.

Step 3, the system recommends relevant data analytics solutions based on the user's data. Step 4, a user activate one or multiple solutions for his/her data analytics. Step 5, the *Cloud Workflow Management* performs cloud-based data analytics in a parallel and elastic manner. Step 6, the final analytics results are returned to the user.

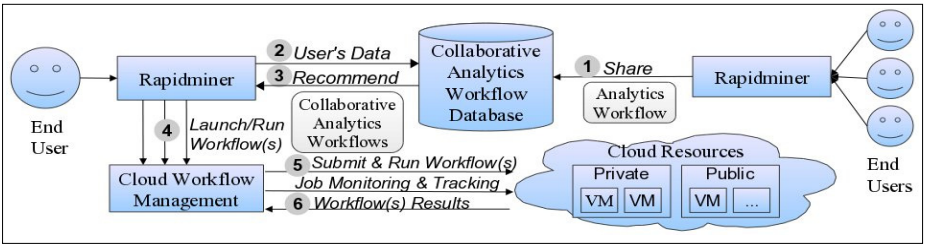


Fig. 1. Cloud-based workflow management solution for collaborative analytics

To handle the collaborative analytics computation resources requirements, we provide a cloud workflow management solution. Cloud workflow management is responsible to schedule and provision the computation resources to execute the recommended analytics workflow efficiently. This solution also support high throughput data processing by running multiple analytics workflows concurrently, high availability through fault-tolerance, and transparency through monitoring and tracking of workflow status. The cloud workflow management encapsulates the complexity of running distributed analytics workflows in a cloud environment.

Our current prototype implementation is built on top of existing analytics workflow suite called RapidMiner [1]. We designed and incorporated new features into RapidMiner to share and recommend analytics workflows based on user's data. Specifically for the recommendation, user's data and the data stored in the collaborative analytics workflow database are characterized based on their statistical properties [2]. The data characteristics of the analytics workflows which are being recommended to the user are similar to the user's data.

2 Demonstration

Our demonstration will showcase two collaborative analytics workflows that we have deployed in our cloud test-bed. The first one is the data mining analysis which extracts statistical pattern from the user data, and the second one is the climate change analysis which analyzes and forecast the weather conditions. These collaborative analytics workflows run in our cloud environment powered by twenty-four cores cluster with hybrid heuristic for scheduling data analytics workflow applications [3].

The data mining analysis demonstrates the sharing of the data mining workflow design, the recommendation of the top three best-suited workflow designs based on the user's data and the workflows execution in the cloud environment. It is a generic use case of collaborative analytics which can be applied to various fields of study.

The climate change analysis demonstrates the uses of collaborative analytics for computational and data-intensive spatio-temporal analysis. It shows the feasibility to deal with spatial and temporal data (i.e. transportation, logistics, geodetics). It supports various data visualisation tools to visualize the analytic results, for example ncview application for thematic data, interactive heat map across time, episodic and trajectory values (i.e., high pressure cells and low pressure centre trajectories).

References

1. Rapidminer – Analytical ETL, Data Mining, and Predictive Reporting, http://rapid-i.com/
2. Castiello, C., Castellano, G., Fanelli, A.M.: Meta-data: Characterization of Input Features for Meta-learning. In: Torra, V., Narukawa, Y., Miyamoto, S. (eds.) MDAI 2005. LNCS (LNAI), vol. 3558, pp. 457–468. Springer, Heidelberg (2005)
3. Rahman, M., Li, X., Palit, H.: Hybrid Heuristic for Scheduling Data Analytics Workflow Applications in Hybrid Cloud Environment. In: Proc. High-Performance Grid and Cloud Computing Workshop 2011, USA, May 16-20 (2011)

Analyzing QoS for Web Service Compositions by QoSDIST

Huiyuan Zheng, Jian Yang, and Weiliang Zhao

Department of Computing, Macquarie University, Australia
{huiyuan.zheng,jian.yang,weiliang.zhao}@mq.edu.au

Quality of Service (QoS) analysis and prediction for Web service compositions is an important and challenging issue in distributed computing. In existing work, QoS for service compositions is either calculated based on constant QoS values or simulated based on probabilistic QoS distributions of component services. Simulation method is time consuming and can not be used in real-time applications for dynamic Web service compositions. Experimental results in [5] show that our proposed QoS calculation approach significantly improves the efficiency in QoS estimation when the QoS of component Web services are probability distributions. In this paper, we present a tool QoSDIST to analyze the QoS for Web service compositions based on our proposed QoS calculation method in [5]. QoSDIST does not put any constraints on the modeling of the QoS of component Web services, i.e., the QoS of a component Web service can be in single value, discrete values with frequencies, standard statistical distribution, or any general distribution regardless of its shape.

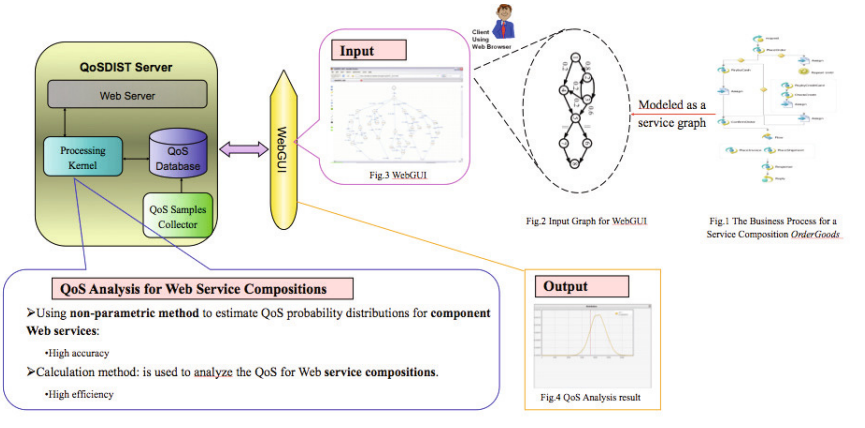


Fig. 1. System Architecture of QoSDIST

Figure 1 shows the system architecture of QoSDIST. Estimating the QoS distributions of a service composition involves a large amount of computing. Therefore, QoSDIST adopts the AJAX client-server architecture. The server is in charge of most of the processing. A Web-based graphical user interface

G. Pallis et al. (Eds.): ICSOC 2011, LNCS 7221, pp. 246–247, 2012.

(WebGUI) runs on the client side which is based on JavaScript[1]. The input information from WebGUI is encoded into XML by the clients and then sent to the server side. On receiving the XML document, the server side will start processing it and return the processing result in XML. QoSDIST involves four major modules: a WebGUI, a QoS Collector, a QoS Database, and a Processing Kernel. The functions of these modules are as follows:

`WebGUI`: *WebGUI* is on the client end of QoSDIST. Through *WebGUI*, a user can input the service graph [6] of a Web service composition and the QoS information for the component Web services in the service composition. The input information will be encoded into XML format and transmitted to the server side of QoSDIST. The QoS analysis result for a service composition will also be output through *WebGUI*.

`QoS Collector`: *QoS collector* is in charge of collecting QoS data for Web services by testing the QoS of Web services at a regular time interval [3]. These collected QoS data for per QoS metric of a Web service is referred to as a QoS sample for that QoS metric of the Web service. QoS samples collected by *QoS Collector* will be transformed into probability distributions based on the method proposed in [5] by *Processing Kernel* and stored at *QoS Database*.

`QoS Database`: The QoS of component Web services are stored at *QoS Database*. The QoS can be in single value, discrete values with frequencies, or probability distributions.

`Processing Kernel`: *Processing Kernel* generates the probability distributions for component Web services based on the samples collected by *QoS Collector* [4]; decodes the received XML information from the client side and gets a graph structure for the service composition; analyzes the QoS of the service composition based on the method proposed in [5]; outputs the QoS analysis result to *WebGUI*.

In this demonstration, we will show how to use QoSDIST to draw the service graph of a Web service composition, edit the properties of the components in the service graph, run the QoS analysis, and read the QoS analysis results for Web service compositions. A screencast video of the demonstration is available at online demonstration for ICSOC2011.

References

1. `http://www.jgraph.com/mxgraph.html`
2. `http://code.google.com/p/flot/`
3. Rosenberg, F., Platzer, C., Dustdar, S.: Bootstrapping performance and dependability attributes of web services. In: ICWS, pp. 205–212 (2006)
4. Zheng, H., Yang, J., Zhao, W.: Qos probability distribution estimation for web services and service compositions. In: SOCA (2010)
5. Zheng, H., Yang, J., Zhao, W., Bouguettaya, A.: QoS Analysis for Web Service Compositions Based on Probabilistic QoS. In: Kappel, G., Maamar, Z., Motahari-Nezhad, H.R. (eds.) ICSOC 2011. LNCS, vol. 7084, pp. 47–61. Springer, Heidelberg (2011)
6. Zheng, H., Zhao, W., Yang, J., Bouguettaya, A.: Qos analysis for web service composition. In: SCC, pp. 235–242 (2009)

[1] mxGraph[1] and flot[2] libraries are used.

A Cloud Resource Orchestration Framework for Simplifying the Management of Web Applications

Rajiv Ranjan[1,2], Boualem Benatallah[1], and Mingyi Wang[1]

[1] School of Computer Science and Engineering, University of New South Wales
{rajiv,b.benatallah,mingyi.wang1}@unsw.edu.au
[2]Information Engineering Lab, CSIRO ICT Centre, Canberra
{rajiv.ranjan}@csiro.au

Cloud computing paradigm [1] has shifted the computing from physical hardware- and locally managed software-enabled platforms to virtualized cloud-hosted services. Cloud computing assembles large networks of virtualized services: hardware resources (CPU, storage, and network) and software resources (e.g., databases, load-balancers, monitoring systems, etc.). Key issue in exploiting the potential of cloud computing is "*Resource Orchestration*". Resource orchestration process spans across a range of operations from selection, assembly, and deployment of resources to monitoring their run-time performance statistics (e.g. load, availability, throughput, utilization, etc.). The process aims to ensure achievement of fault-tolerant and QoS fulfillment states by resources and applications through adaptive management.

Existing cloud resource orchestration techniques require human familiarity with different types of resources and typically rely on procedural programming in general-purpose or scripting languages. The interaction with resources is mainly performed through low-level APIs and command line interfaces. Given the proliferation of new providers offering resources at different layers (e.g. software-as-a-service, platform-as-a-service, and infrastructure-as-a-service), such orchestration techniques are therefore inadequate to make the cloud resources accessible to a wide variety of users, particularly from non-IT domain.

To improve this situation, an innovative Resource Orchestration Framework (ROF) is presented in this demonstration. The ROF leverages java widget programming, virtualization platforms and Amazon's open-source APIs [2] to enable simplified, intuitive resource orchestration and web application management. The ROF allows users to graphically browse available resources, Virtual Machine (VM) images, storage repository, application hosting environment, and application components. Further, it supports deployment, configuration, and monitoring of resources and applications directly from Java widgets.

The high level architecture, as shown in Fig. 1 (a), consists of two layers: *Programming Layer:* implements the business logic for the interface exposed by widgets. Programming layer is designed based on an extensible software engineering pattern that allows to plug-in service APIs of different clouds. Though our current implementation works only with Amazon EC2 cloud, it can be easily extended to support other clouds. *Widget Layer:* encapsulates user interface components in the form of six principle widgets including: VM, Instance, Storage, Monitor, Application, and Security. All the authentication and authorization credentials (e.g. secret key, SSH key pairs, security group, etc.) are managed by the Security Widget. On the other hand, Storage widget allows users to upload Java Web Application Archives, database

G. Pallis et al. (Eds.): ICSOC 2011, LNCS 7221, pp. 248–249, 2012.

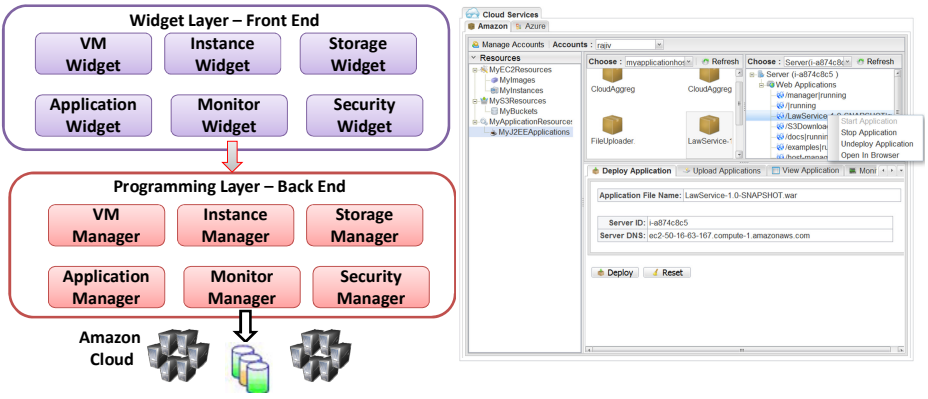


Fig. 1. (a) ROF Architecture. (b) A screenshot of application widget.

schema definition files, and VM images to S3. The basic configuration of VM images including their architecture, imageId, state, and virtualization platform can be viewed through the VM widget. For configuring the deployment (e.g. starting, stopping, and terminating), security credentials, and monitoring preference of VM images, users need to interact with the Instance widget.

The functionality for launching, deploying, and monitoring individual application components and hosting environments is made feasible by Application widget as shown in Fig. 1 (b). This widget also features an inline browser where hosted web application can be actively viewed and tested. The main task of the Monitor widget is to monitor the run-time status of VM instances, network, storage resources and application components. For viewing the performance statistics on per web application basis, one can switch to the Monitor widget.

To our knowledge, no related cloud ROF [2, 3, 4] supports: (i) inline browsing of the hosted web application; (ii) dragging & dropping the application components (e.g. JWARs and HyperSQL [5] DB files) form cloud storage (e.g. S3) to hosting environment (e.g. Tomcat and HyperSQL DB); and (iii) controlling lifecycle activities (e.g. start, stop, refresh, and undeployment) of application components and hosting environments via the widget-level interface.

The purpose of this demonstration is to showcase the effectiveness of ROF in simplifying the process of resource orchestration and web application management. The demonstration will show that prior knowledge of existing cloud resource orchestration tools and concepts is not mandatory for users. Finally, the demonstration will utilize multi-tier web application as the application scenario. It will be deployed within the Tomcat container and its database layer will be managed by HyperSQL [5] database. ROF's screenshots can be found at: http://rranjans.wordpress.com/c-tool.

References

1. Armbrust, M., et al.: A view of Cloud Computing. Communications of the ACM Magazine 53(4), 50–58 (2010), doi:10.1145/1721654.1721672
2. Amazon Web Services, `http://aws.amazon.com/` (accessed September 2011)
3. Rightscale Cloud Framework, `http://www.rightscale.com/` (accessed September 2011)
4. Cloudswitch Framework, `http://www.cloudswitch.com/` (accessed September 2011)
5. HyperSQL Database, `http://hsqldb.org/` (Accessed September 2011)

A Tool Suite to Model Service Variability and Resolve It Based on Stakeholder Preferences

Erik Wittern[1], Christian Zirpins[1], Nidhi Rajshree[2], Anshu N. Jain[2], Ilias Spais[3], and Konstantinos Giannakakis[3]

[1] eOrganization Group, Karlsruhe Institute of Technology (KIT)
[2] IBM Research India
[3] Athens Technology Center S.A. (ATC)
{erik.wittern,christian.zirpins}@kit.edu
{nidhi.rajshree,anshu.jain}@in.ibm.com
{i.spais,k.giannakakis}@atc.gr

Abstract. Modern information and communication technology creates new possibilities to enable participative, collaborative service design. We present a novel approach of integrating stakeholder preferences into service design. Our approach is based on modeling multiple service configurations, thus including variability into the service design. Stakeholders can evaluate value-relevant service aspects on a web-based deliberation platform. Ultimately, all stakeholder opinions are aggregated to hint the service engineer on the preferred way to implement the service. In this demo we present our tool suite to model services and their variability and the server that handles the opinions stated on the deliberation platform.

Keywords: Service Variability Modeling, Participatory Design, Preferences.

1 Introduction

Participative service design promises valuable input for and increased acceptance of the implemented service. Especially the design of public services, that have tight budget constraints and very large target groups, profits from integrating diverse stakeholder opinions. The COCKPIT project [1] aims to develop a new methodology and corresponding tool suite, shown in this demo, to enable specifically citizens' participation in the design and re-design of public services.

2 Concept, Implementation and Benefits of the Tool Suite

Our tool suite includes the *public service modeler* to capture generic information about the public service, for example stakeholders, requirements or legal information. Additionally, a process modeling component is included to capture the to-be process of the public service. The *Service Feature Model designer (SFM designer)* models variability of a service using *service feature modeling* [2]. *Service*

G. Pallis et al. (Eds.): ICSOC 2011, LNCS 7221, pp. 250–251, 2012.

features represent alternative implementation aspects of the service, for example process elements or resources. Based on a complete service feature model, the *valuation server* automatically constructs an *evaluation* and publishes it on the web-based *deliberation platform*. In a *poll*, stakeholders *vote* on their preferences regarding *attribute types* that describe value-relevant aspects of the service features. The valuation server assesses the vote and immediately feeds back the *service configuration* that ideally matches the stated preferences. At the end of the evaluation, an aggregated preference is provided to continue the service design under consideration of all collected opinions. The great benefit of our tool suite is that it provides an integrated approach wherein the collected stakeholder opinions are directly matched to service design choices.

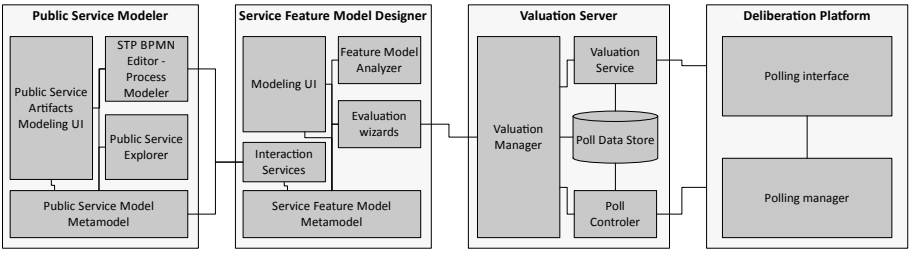


Fig. 1. Architecture overview of the tool suite

Figure 1 shows the tool suite's components and their relationships. REST interfaces realize the interaction between the valuation server and the other components.

3 Demonstration Overview

The demonstration shows how the overall toolkit can be utilized to support stakeholder integration in the modeling of a scenario from the COCKPIT project. We provide the screen case online[1].

Acknowledgments. This work was supported by the COCKPIT project [1].

References

1. COCKPIT Project: Citizens Collaboration and Co-creation in public Service Delivery (2010), http://www.cockpit-project.eu
2. Wittern, E., Zirpins, C.: On the Use of Feature Models for Service Design: The Case of Value Representation. In: Cezon, M., Wolfsthal, Y. (eds.) ServiceWave 2010 Workshops. LNCS, vol. 6569, pp. 110–118. Springer, Heidelberg (2011)

[1] http://www.youtube.com/watch?v=Nyodp4dJk4U

CAptEvo: Context-Aware Adaptation and Evolution of Business Processes*

Antonio Bucchiarone, Annapaola Marconi, Marco Pistore, and Heorhi Raik

FBK-Irst, via Sommarive 18, 38123, Trento, Italy
{bucchiarone,marconi,pistore,raik}@fbk.eu

Abstract. CAptEvo is a framework for the adaptation and evolution of service-based business processes operating in dynamic execution environments. In this demonstration, we apply the CAptEvo to a case study from the logistics domain and show its advantages in handling highly complex dynamic real-world business applications.

1 The CAptEvo Framework

Adaptability is a key problem in dynamic business environments, where operational excellence requires to model and execute business processes taking into account a dynamic, open and non-deterministic execution context. These adaptation needs may be triggered by specific execution cases, dynamic service availability, non-controllable situations depending on environmental conditions, or changing requirements. Moreover, this need for continuous adaptation results in a system characterized by a huge set of process executions that, although instantiated on the same process model, strongly differ in terms of process structure. In such a dynamic environment, the *short-term* adaptations applied to process instances should be used to derive *long-term* changes that progressively improve the process models.

The CAptEvo Framework, developed within the ASTRO project [1], integrates sophisticated techniques for managing the execution, adaptation, and evolution of context-aware business processes. The framework exploits a modeling approach for Service Based Applications where adaptability and context-awareness are key embedded characteristics of the business application [2,3]. During the execution phase, process models are instantiated and the corresponding process instances are executed. The *Execution Manager* is responsible for keeping the system configuration up to date and for consistently aligning the status of context properties to the execution of the processes and to the context events. The system configuration is used by the Execution Manager to monitor context constraints associated to running process instances and to trigger adaptation in case of violation. The *Adaptation Manager* supports two types of AI-planning based dynamic adaptation: *vertical* and *horizontal*. The aim of vertical adaptation is to refine abstract activities of a process by automatically composing

* The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement 215483. (Network of Excellence S-Cube).

[1] www.astroproject.org

G. Pallis et al. (Eds.): ICSOC 2011, LNCS 7221, pp. 252–254, 2012.

available services and obtaining a concrete process that can be executed. Horizontal adaptation results in a structural modification of the process instance by adding, changing or removing process activities to retain the reachability of its original goals in case of a changing environment. The information about the system execution and adaptation is recorded in the *Execution Log*. Examples of stored information are the traces of process instances and of context properties execution and the adaptation history in terms of adaptation problems and adopted adaptation variant. The set of adapted process instances together with the information concerning their execution are used as training cases for evolution mechanisms in order to progressively improve process models that are then used to instantiate future process instances. The evolution need is triggered by a problem in the system performance with respect to the KPIs of the process models. Given the specific evolution need (KPI violation for a certain process model [5]), the Analyzer considers all the process instances that contributed to the KPI violation and looks for recurring adaptation needs examining their adaptation history. The *Evolution Manager* looks for adaptation variants that solve the same adaptation problem and that have good performance with respect to the KPI. The identified adaptation variants are then ranked according to execution performance (e.g., performance with respect to the other KPIs and confidence) and proposed as evolution variants to be plugged-in in the original process model. In our framework we adopt a man-in-the-loop approach, where evolution variants, together with their performance, are presented to the process designer that, through a set of *built-in adaptation tools* [4], can embed them in the evolved process.

2 Demonstration

To demonstrate the CAptEvo framework in action, we use a real-world scenario from the domain of logistics [1]. The scenario is based on business processes used in the terminal of the Bremerhaven sea port, where cars arriving by ship have to be delivered to retailers. Before the cars can be delivered, a series of activities needs to be completed such as customization procedures, car shipment and repair, etc. The management of car delivery is a highly complex process, as each car requires an individual treatment, and the process execution might be affected by changes in the execution context such as car damages. This requires sophisticated modeling that allows for run-time adaptation, and evolution of the application. In our demonstration, we illustrate the CAptEvo framework in action and present the outcome of our algorithms to the end users. We have created a visualization environment enabling interaction between the framework and the user and simulating execution, adaptation and evolution of business processes in our case study. In particular, it can:

- Run the reference "Car Logistics" scenario and simulate the execution and adaptation of each business process attached to each car.
- View the different adaptation strategies supported by our framework (i.e., vertical and horizontal) and how they are used during the scenario execution.

- Inspect the behavior of the system in terms of process performance and adaptation history.
- View the process evolution results and choose the process variants to be embedded in the system.

The goal of this demonstration is to show the novel concepts and advantages of the CAptEvo Framework when applied to a real and complex pervasive system.

References

1. Böse, F., Piotrowski, J.: Autonomously controlled storage management in vehicle logistics - applications of RFID and mobile computing systems. International Journal of RT Technologies: Research an Application 1(1), 57–76 (2009)
2. Bucchiarone, A., Kazhamiakin, R., Pistore, M., Raik, H.: Adaptation of Service-based Business Processes by Context-Aware Replanning. In: SOCA 2011 (2011) (submitted)
3. Bucchiarone, A., Marconi, A., Pistore, M., Sirbu, A.: A context-aware framework for business processes evolution. In: EVL-BP Workshop of EDOC 2011 (2011)
4. Marconi, A., Pistore, M., Sirbu, A., Eberle, H., Leymann, F., Unger, T.: Enabling Adaptation of Pervasive Flows: Built-in Contextual Adaptation. In: Baresi, L., Chi, C.-H., Suzuki, J. (eds.) ICSOC-ServiceWave 2009. LNCS, vol. 5900, pp. 445–454. Springer, Heidelberg (2009)
5. Wetzstein, B., Leitner, P., Rosenberg, F., Dustdar, S., Leymann, F.: Identifying influential factors of business process performance using dependency analysis. Enterprise IS 5(1), 79–98 (2011)

A Registry and Repository System Supporting Cloud Application Platform Governance

Dimitrios Kourtesis and Iraklis Paraskakis

South-East European Research Centre (SEERC), City College - International Faculty of the University of Sheffield, Proxenou Koromila 24, 54622, Thessaloniki, Greece
{dkourtesis,iparaskakis}@seerc.org

Abstract. The CAST project is putting forward a novel ecosystem-oriented model for developing and deploying enterprise applications on future cloud application platforms. One of the most fundamental challenges that this model is raising concerns how to support platform governance. In this demonstration, we present the registry and repository solution that was created to address the platform's requirements with respect to centrally-exercised management and quality assurance.

Keywords: Cloud Application Platform, Platform as a Service, PaaS, Governance, Registry and Repository System.

1 Introduction

The cloud computing paradigm is presenting new challenges and opportunities for the world of enterprise computing. Cloud application platforms, in the form of various types of commercial Platform as a Service (PaaS) offerings, are expected to transform the way enterprise software will be developed and provisioned in the years to come.

The CAST project[1] is a collaborative EU-funded research effort which is set to investigate the challenges associated with engineering such software platforms. The aim of the project is to develop a cloud application platform that not only supports the development and deployment of on-demand (SaaS) enterprise software applications, but does so in a way that facilitates the creation of a value network and ecosystem around the platform. Instrumental in achieving such an effect is to employ a development and deployment model that promotes collaboration and reuse of resources. In the CAST platform, in addition to the option of coding a solution from the ground up, developers are given the option to create their solutions by reusing, adapting and combining third-party apps that are already deployed to the platform, as well as integrating their apps with externally deployed systems via Web services [1].

Developing the platform infrastructure to support such an ecosystem-oriented development and deployment model entails several challenges. Taking into consideration the increasing numbers of diverse and interdependent solutions,

[1] http://www.cast-project.eu

G. Pallis et al. (Eds.): ICSOC 2011, LNCS 7221, pp. 255–256, 2012.

appsand external services that will accumulate on the platform as time advances, governance becomes one of the most fundamental requirements to be addressed.

In the CAST project, this requirement was addressed by transferring some best practices and approaches from the field of SOA governance, i.e. by realising centrally-exercised management and quality assurance through a registry and repository system.

2 System Overview

The registry & repository system of the CAST platform serves as a central location in which entities and artefacts that are necessary to the operation of the platform are stored, organised, and managed throughout their lifecycle. It provides a space and a set of functions for enabling governance of entities and artefacts from creation to retirement. Governance is supported through tools which assist the users of the system (i.e. platform administrators and solution developers) in performing manual quality assurance tasks, but also, through tools that automate a wide range of quality controls, applying conformance checks and validations with regard to platform governance rules. The registry & repository system was developed upon the open source WSO2 Governance Registry project.

3 Functions/Features to be Demonstrated

The main functions of the registry & repository system that will be demonstrated are:

- Cataloguing and storage: Platform solutions, apps, and services are catalogued and their associated artefacts stored in a central location
- Policy conformance checking: Managed entities and their artefacts are checked for conformance to a range of platform policies
- Lifecycle management: The evolution of managed entities follows an explicitly defined lifecycle model, where transitions to states are guarded by preconditions
- Dependency tracking and impact analysis: Dependencies among solutions, apps, and services are tracked to allow for impact analysis
- External service monitoring: External services on which apps are depending are monitored to ensure appropriate levels of availability and responsiveness

Acknowledgments. The CAST project is co-funded by Eureka Eurostars (E! 4373).

Reference

1. Kourtesis, D., Kuttruff, V., Paraskakis, I.: Optimising Development and Deployment of Enterprise Software Applications on PaaS: The CAST Project. In: Cezon, M., Wolfsthal, Y. (eds.) ServiceWave 2010 Workshops. LNCS, vol. 6569, pp. 14–25. Springer, Heidelberg (2011)

Integrated Asset Analysis Framework for Model-Driven Development of SOA Based Solutions

Karthikeyan Ponnalagu, Nanjangud C. Narendra, and G.R. Gangadharan

IBM Research India, Bangalore, India
{pkarthik,narendra,gangadharan}@in.ibm.com

Abstract. In SOA based application development, a plethora of architectural constructs such as processes, services and components need to be built. This requires modeling of the application at different levels of abstraction such as business architecture, application architecture and runtime architecture. Model driven development (MDD) is hence considered the primary development approach for building SOA applications. Existing MDD methodologies and tools only support searching and discovery of assets, and do not support their analysis in order to determine their suitability for reuse. This often results in selecting potentially incompatible assets among the various layers of the solution, resulting in redundant asset customizations. In order to address this issue, we present a novel framework and methodology that enables the integrated analysis of existing assets associated across multiple abstractions of the solution from different asset repositories. This approach helps in creating a consistent asset reusability view across all the phases of SOA development with multiple reusable asset options to compare and select. We present an experimental evaluation of our methodology on real-life SOA assets distributed across multiple repositories and illustrate how our integrated mechanism can help consistently maximize reuse of assets in SOA development.

Keywords: Service-Oriented Architecture, Asset Analysis, Reuse.

1 Introduction

In SOA based solution development, the Model Driven Development (MDD) approach enables business analysts and application architects to better integrate business requirements with IT specifications [1,2,3,4]. As illustrated in Figure 1, in a typical SOA based solution development, models at higher abstraction level representing the business domain are translated into service models, which in turn are refined and realized as traditional design models (class diagrams, sequence diagrams, etc.) finally followed by implementation. The process of discovering and leveraging existing potential reusable assets from repositories will minimize the cost and effort of building SOA based solutions, rather than developing from scratch. However, improper selection of reusable assets across the various layers of the solution results in incompatibilities among the assets, thereby incurring increased customization effort. As the volume of available assets in most

G. Pallis et al. (Eds.): ICSOC 2011, LNCS 7221, pp. 257–269, 2012.

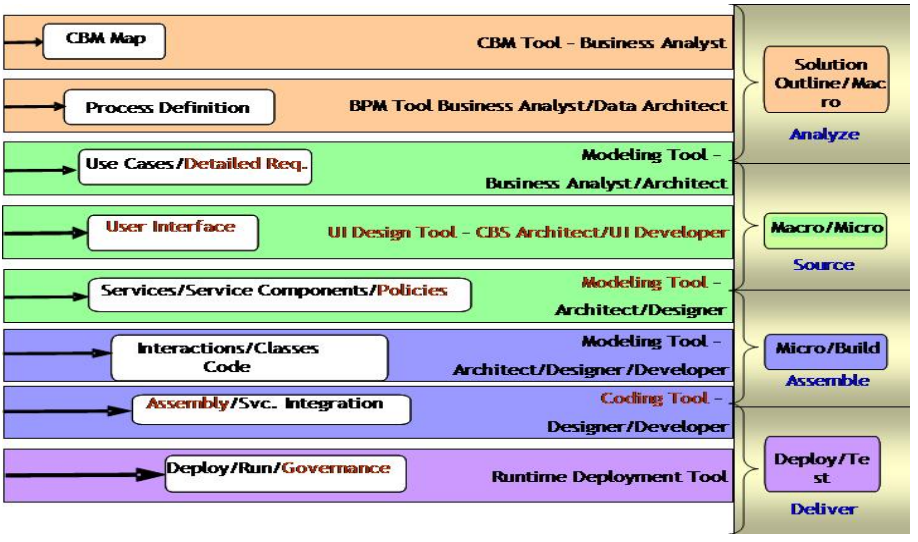


Fig. 1. SOA Lifecycle Tooling View

repositories are generally large (in terms of hundreds), discovering the appropriate asset and customizing it for compatibility with the existing artifacts in the SOA based solution is quite challenging. This exercise can even take up to a week per asset for an experienced architect.

Therefore, an integrated view of assets from multiple abstractions sourced from different repositories at different stages of development is required. Such a view enables architects to identify the most appropriate set of assets to reuse with minimal customization effort. This kind of view can also be reused for similar solutions in the same domain reducing the time invested in searching for the assets in the first place. In this perspective, the key contribution of our paper is an integrated asset discovery and analysis framework to aid model-driven development of SOA based solutions. The salient features of our paper are as follows:

- A novel approach that progressively generates a reusable integrated asset analysis reference model from multiple repositories across the different abstraction layers of the solution
- An extensible mechanism that supports diverse abstractions of business and service models towards integrating newer asset specifications and repositories
- An integrated asset assessment that enables both top-down and bottom-up trade-off considerations in selecting or rejecting candidate assets
- A mechanism that enables dynamic selection of search context related filters discovered from the initial searches and employed in subsequent searches

The remainder of the paper is organized as follows. We present a Request for Quote (RFQ) process model as our running scenario in Section 2. Section 3 discusses the different architectural layers of the SOA model and corresponding support for asset analysis across different asset specifications and repositories. In

Section 4, we present our framework and methodology for analyzing existing architectural assets in SOA based solution design. Section 5 illustrates asset search and identification algorithm that analyzes existing assets across multiple repositories and helps in identification of reusable assets via a context-based filtering mechanism. In Section 6 we evaluate our framework and its constituent algorithm on our running example. Related work is discussed in Section 7, followed by concluding remarks in Section 8.

2 Running Example

Request for Quote (RFQ) business process provides a facility for automating the premium quote generation for customers who wish to obtain insurance policies in various sectors including automobiles, home, and life.

Nowadays independent search, discovery, and reuse assessment is conducted across each of the phases with differing asset requirements. Similarly, we perform independent asset searches with a random sequence for specific asset types in the RFQ application. A total of 33 independent searches are conducted through a single user interface connected with multiple repositories (see Section 6 for more details). Each search is focused on searching a specific asset type ignoring the search results from the previously executed search. Thus, a total of 182 assets are discovered. Figure 2 depicts the distribution of these assets across the different modeling abstractions.

We observe that the discovered assets have reuse compatibility issues with each other because they do not share the same characteristics such as status of their approval, supported languages, and compliance to standards or regulations. Some of these assets have high mismatch probabilities if they are considered for reuse with assets from other asset types. For example, the third search in the independent search sequence is based on case study as asset type. From the total of 8 filters associated with the assets resulted in this search, 6 filters have unique values leading to a mismatch probability of 75%. We illustrate the associated mismatch probabilities for assets belonging to all the asset types in Figure 2. Thus we have a larger scale of asset types to consider for reuse in each of the solution development phases. Such a scenario leads to redundant customizations and inconsistencies in solution development due to the lack of a methodology that integrates the asset search and discovery steps and helps in generating a consistent system-wide asset reuse view. Figure 3 illustrates the RFQ business process retrieved from one of the searches as a process map asset type belonging to business architecture layer.

3 Exploring Asset Analysis in SOA Based Solutions

Understanding the architecture and the associated models of an SOA based solution is paramount in setting the context for search, discovery and analysis of existing assets. From our experience in analyzing SOA based solutions, we have noticed that there are multiple touch points to initiate asset search. We describe

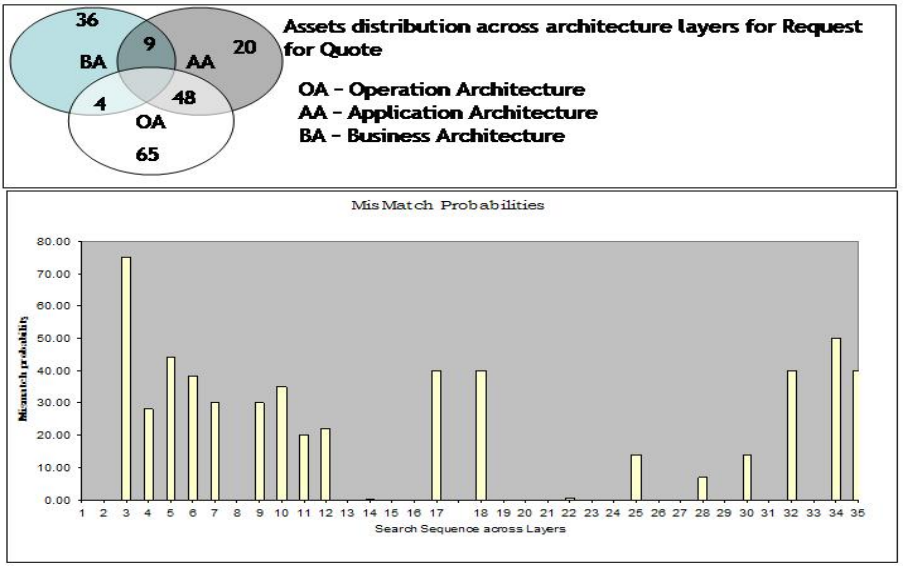


Fig. 2. Asset Distribution

them under the following three layers of abstraction: Business architecture, Application architecture, and Operation architecture.

In the context of SOA, business architecture models involve business process models, component business models, requirement models, and domain data models. The most common and widely used Business Architecture specification is the business process model as discussed in [5]. The Component Business Model (CBM) [6], a methodology for business architecture modeling defines a hierarchy of business components, helps in identifying the domains the assets can tentatively belong to, while the process model focuses on validating the functional specifications of the discovered assets. Thus the impact of discovering and analyzing existing assets in the business architecture space is much higher, because of the context of information available at the business specification level and the flexibility one can assume with the higher abstraction the associated models bring in. However there should be no conflicts in integrating the business architecture assets with subsequent selection of application and operational assets.

Application architecture methodologies provide a mature environment for architecting and designing SOA solutions. The core functionality of such methodologies includes supporting multiple business specification models and enabling architects to develop the correct SOA based design of the solution. They also provide capabilities to represent business architecture specifications such as process models in the form of UML Models. From a MDD perspective, they enable generation of multiple implementation models depending on the stipulated deployment environment.

Operational architecture models basically involve the enhancement of generated implementation artifacts for hosting in a runtime environment, focusing on

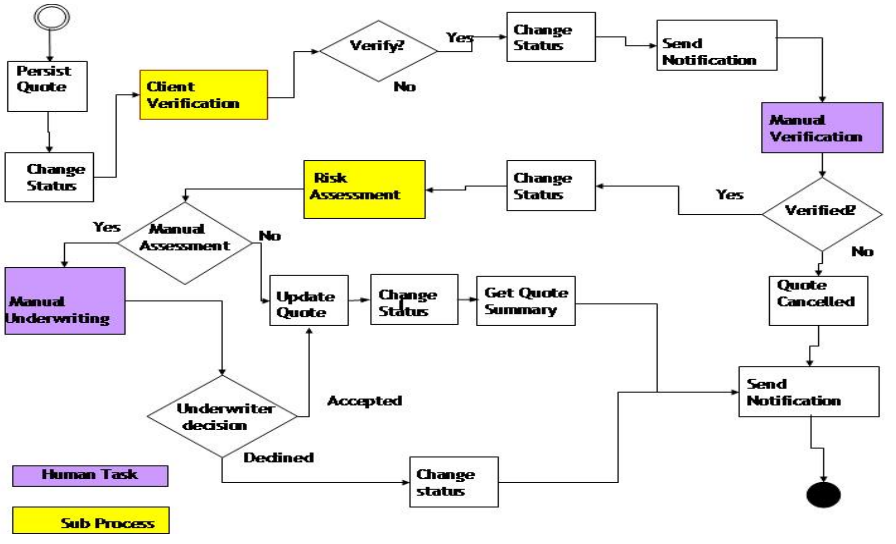


Fig. 3. Process Model Representation for the Request for Quote Solution

actual custom development and generation of exclusive set of artifacts for the runtime environment. This is supported by model driven engineering capabilities that generate starter code at the completion of the solution design exercise. Hence, operational architecture for SOA based solutions provide greater scope of code reusability especially with the MDD approach.

An integrated view of associated models related with RFQ application for these three layers is illustrated in Figure 4. Due to the lack of integrated and implicit automated asset analysis and discovery mechanisms in typical MDD approaches, there exists such a closely linked chain of capabilities that quickly model and transform a business specification model into an application architecture specification and associated set of code artifacts.

4 Asset Analysis Framework: Design and Methodology

Our framework complements the service specification and design components prevalent in MDD tools such as Rational Software Architect (RSA)[1] via discovery, selection, and adaptation of existing assets that can realize the identified service functionalities without fresh implementation at multiple SOA layers. In model driven SOA solution context, our framework incorporates a way to access assets stored in different repositories and to view existing assets from one integrated modeling tool.

Our framework basically contains three main components (see Figure 5):

1. The Contextual Asset discovery component, that enables the construction of multiple optimized asset search queries based on the business architecture

[1] http://www-01.ibm.com/software/awdtools/swarchitect/websphere/

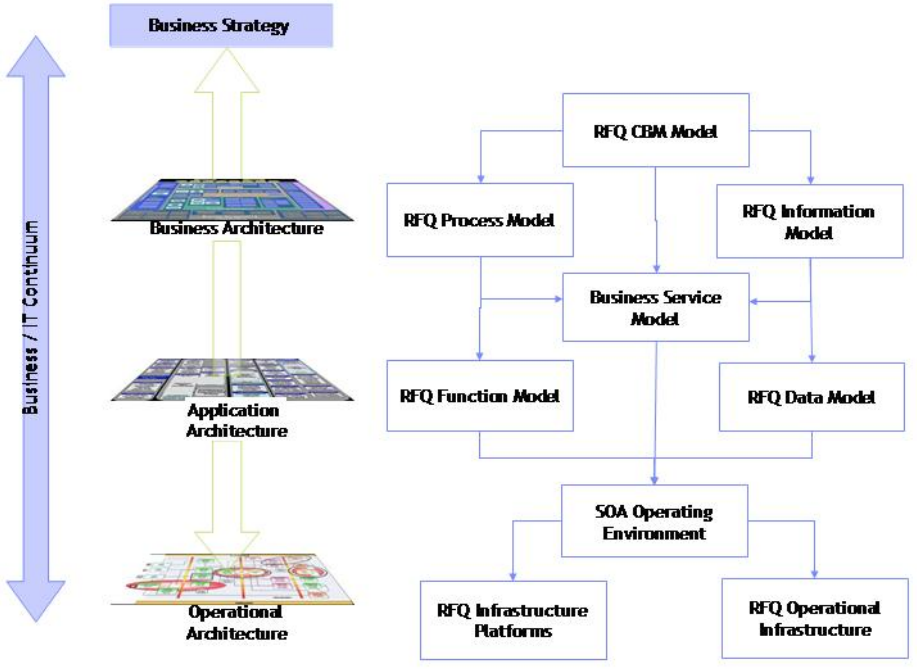


Fig. 4. An integrated view of RFQ Application models

model and to initiate asset search in iterative fashion across the multiple repositories and across multiple abstractions such as service models, designs, service component definitions and code implementations

2. The Core Asset Modeling component that retrieves the asset artifacts and represents them in the generic Asset Model.
3. The Asset Analysis Component that facilities automated or manual comparison of the selected asset models against the corresponding business architecture elements such as business process, business entity models

The requirement-centric context used for initiating search and selection of filters is formulated at two levels. The initial level is based on the type of business specification document such as process model, CBM, business use case model, etc. This basically helps in identifying the type of artifacts such as service model, detailed design, actual code implementation, deployment plan, etc., that will be required to take the solution development to its completion. The second level is based on the actual structural and behavioral content (such as the details of tasks, sub-processes, etc.) that dictates the search of functional and technical specification of the asset artifacts.

The metamodel illustrated in Figure 6 contains the UML based representation of the key asset types. Transformation of asset types and the associated structural constructs by the assets into UML metamodel helps in performing a uniform analysis with respect to the given solution requirement.

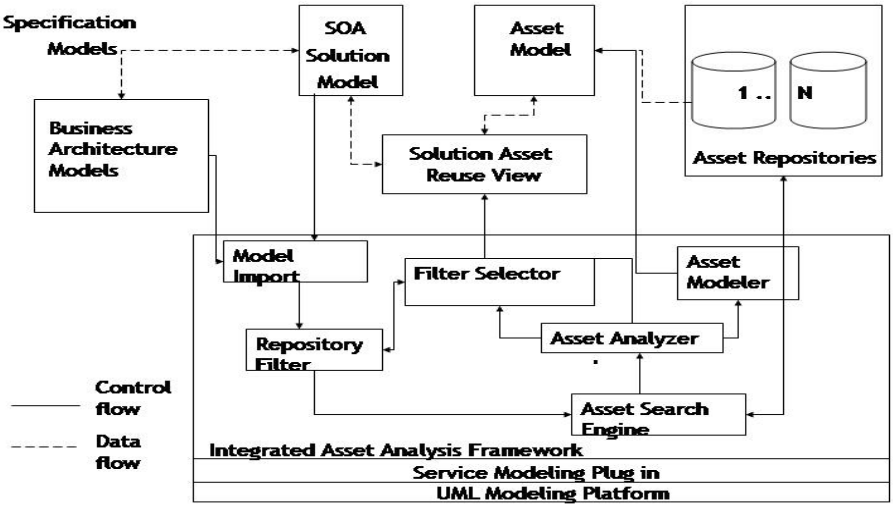


Fig. 5. Asset Analysis Framework Generic Design View

5 Contextual Asset Search and Identification Algorithm

Our novel algorithm for asset search and identification provides greater control for accurate discovery and analysis of existing assets (see Algorithm 1). At the beginning, the framework needs to be registered with the list of repositories prevalently used in the organization. The algorithm basically follows a two-phased search approach. The first phase of our algorithm iterates across all the repositories to find the suitable match for the process or the associated tasks with limited search. The search is initiated with the process related information retrieved from the UML representation and enhanced by adapting the search query based on the predesigned format supported in the repository under search. The results retrieved are consolidated with the removal of duplicate results on subsequent iterations. The assets thus retrieved are associated with their basic information appended to the corresponding elements in the process model. In each repository, a basic validation is performed to ensure whether each of the elements in the process model have at least one asset associated with them. Otherwise the search for unassociated elements are repeated in other repositories.

In most cases, such a basic search results in a huge number of assets with most of them being unrelated or obsolete. Performing a manual validation of the appropriateness of these assets for a solution context is one of the factors that make architects to go for development from scratch. In our earlier work, [7], we defined a business process consisting (a) a set of associated service models, (b) data dependencies between the services based on the execution of preceding services (produced data dependencies), (c) data dependencies between the services based on the input model of preceding services (received data dependencies), and (d) control flow dependencies that provides the choreography of the services. Extending this modeling representation for searching existing assets,

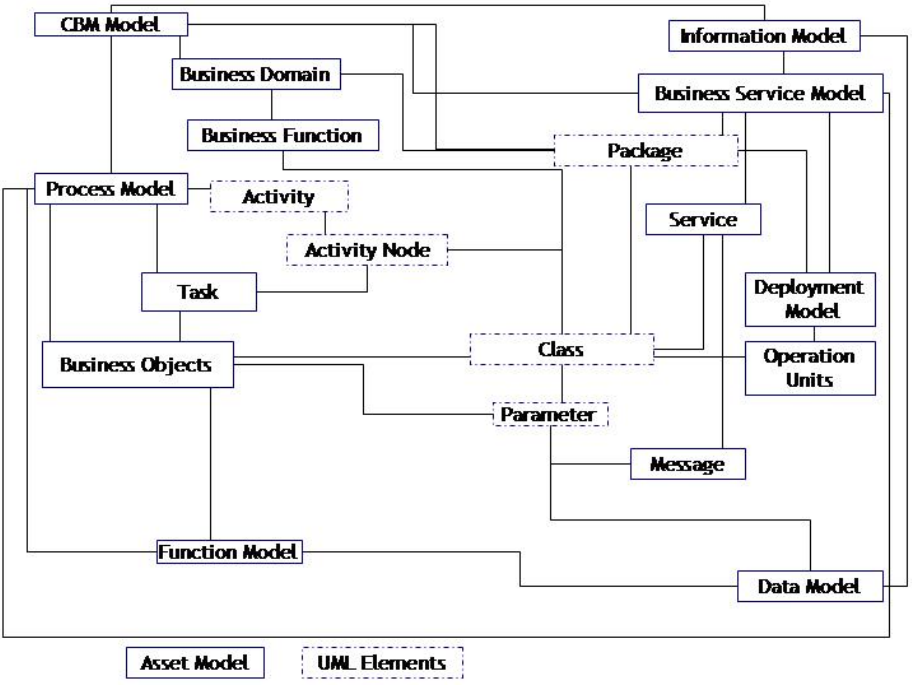


Fig. 6. Asset Analysis Metamodel

the second phase of algorithm proceeds as follows: Task models execute base search aided with the set of input and output data models. The control flow and data flow dependency models categorize search with single repositories and also extract filter matching that reduces the search space with lower number of asset artifacts. Basically filters are multi-dimensional and help us categorize results as per domain, functional, or technology perspective. This enables asset identification that can realize the functionality of a group of tasks with a single coarse-grained asset. This guided search approach will complement work such as [8], where the authors discussed SOA centric transformation of pre-identified legacy assets repositioned as services and processes. Iterating the second phase subsequently for all task groups results in a reduced single asset association for the tasks or the process itself. The actual execution of this algorithm depends on the searching capabilities and metadata support of the specific repositories.

6 Implementation and Evaluation

The prototype of the proposed framework has been developed as a set of plug-ins. The core functionalities such as representation of assets as UML models and analysis of the generated asset model from an imported asset artifact are implemented as a central plug-in. Subsequently for each supported repository, a corresponding repository specific plug-in is developed and integrated into the

Algorithm 1. Integrated Asset Search and Identification Algorithm for a Process Model

Get $P = S, E, D, C$
Get $R_1, \ldots, R_v$
Populate unalloted list of elements $E[]$ with $S_1, \ldots, S_n$ for P
$E[] = E_1 \ldots E_n + 1$
for all $R_i \in R$ **do**
  for all $E[i] \in E[]$ **do**
    Construct base search query for R_i with $E[i]$
    Initiate base search with $E[i]$
    Get ResultSet[] R_p
    ConsolidateSearchResult (R_p,$E[i]$,E)
  end for
  if $E[] = \emptyset$ **then**
    break
  end if
end for
Generate candidate asset model with final result set F
Get asset list $A[]$ from F
if $[A] > [P]$ **then**
  FilterSearch ()
else
  Build solution template with candidate assets
  Proceed to manual asset analysis and new process design
end if

procedure *ConsolidateSearchResult*(*ResultSet*[] *R*, *Element Ei*, *Vector E*)

Identify asset validity for status and accessibility
Identify asset uniqueness (whether they are not already allotted)
Group related assets (through existing relationships with other allotted assets if any)
Assign allocation link with the corresponding E
Remove E from unalloted list $E[]$
Populate the resultant R to F

procedure *FilterSearch*()

Create task groups based on E, D, C from P
for all R_i **do**
  Identify corresponding asset lists A_E, A_D, A_C
  Identify common filters across each of A_E, A_D, A_C
  Apply common filters across the final Asset List $A[]$
  Consolidate result list and remove undiscovered from $A[]$
end for

framework. The functionalities of the repository specific plug-ins involve: connecting to the corresponding repository through an authenticated connection as mandated by the repository; and enabling search, discovery, selection, and retrieval of assets from the repository.

We evaluated the prototype on a SOA design project for developing Request for Quote business process model (illustrated in Section 2). The objective was to identify reusable services in the business domain and to further design and implement them. We have designed the business process model using Websphere Business Modeler [2] and logical Data model in Rational Data architect [3]. Both the models can be imported into Rational Software Architect(RSA) for service identification. The Service Model is built using SOMA-ME [4], which is a service modeling application developed as a RSA plugin. RSA's UML to SOA transformation transforms UML design artifacts to create the code artifacts from the UML Service Model for those elements that are not available as an existing asset from any of the connected repositories. For evaluation, we have integrated the iRAM[4] repository with our framework to export, import, reuse, track, and analyze any type of assets including SOA service model and other related work products. We choose iRAM as the repository for conducting our experiments because it enables search and discovery of multiple asset types from a single web browser interface.

As we discussed in Section 2, the available set of assets based on the search meta-data was quite large, i.e., 182, from the first phase of our algorithm. Many of these assets have high mismatch probabilities if considered for reuse with other independently identified assets. Hence we proceed for further filtering to identify the most appropriate set of assets for building the required RFQ solution. At the completion of each search, we collected the search results and the corresponding filters associated with all of the assets available in the search space. Now we randomly selected any of the assets from the search space and identified the corresponding filters.

As we see in Figure 7, the initial search on just specific asset types is similar to the first phase of our algorithm in terms of total number of search results. Then the subsequent searches based on the retrieved filters are conducted. We observe that the subsequent searches always result in assets in the range of 2-5 (in most of the cases), irrespective of the type of assets we searched in the initial search. Now we compare the independent asset search and integrated asset search corresponding to the asset types "Architecture Documentation" and "Code/Software Components". The independent search would have resulted in 9 assets belonging to the type "Architecture Documentation" and 16 assets belonging to "Code/Software Components". With the integrated search approach, if we started our initial search with "Architecture Documentation" and selected one of the 9 assets, subsequent searches based on the filters associated with the selected asset would not have resulted in any assets. This means selecting any

[2] http://www- 01.ibm.com/software/integration/wbimodeler/
[3] http://www-01.ibm.com/software/data/integration/rda/
[4] http://www-01.ibm.com/software/awdtools/ram/

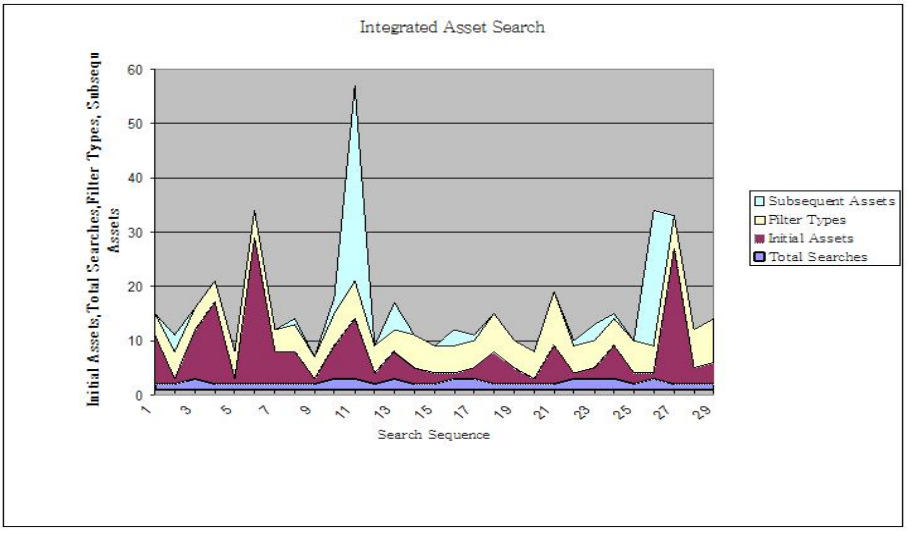


Fig. 7. Integrated Asset Search

other assets belonging to a different asset type purely based on independent searches would have resulted in reuse mismatch. For example, we can compare the redundant exercise involved in selecting and analyzing one of the 16 assets for the type "code/ Software Component" that subsequently results in reuse mismatch with a selected asset of type "Architecture documentation" in the independent search approach. Our integrated approach helps avoiding at least 15 independent searches and subsequent exercise of selecting and analyzing from at least 170 assets. This indicates a scale down from 182 assets to 9 assets to be considered if the initial search is started with the asset type "Architecture documentation". Figure 7 illustrates the integrated search in terms of number of subsequent searches, the total assets resulted from the subsequent searches , with the initial search for each of the asset type.

7 Related Work

Generally, asset analysis in model driven service design and development is seen as a challenging research problem. Rainer et. al. provide an overview of method imperatives that address the requirements for SOA projects [9]. Zimmermann et. al. [10] propose a multi-level SOA decision catalog that includes asset reuse more as part of service realization techniques. [5] presents an overview of the SOA specific repositories currently used in practice across the different phases of the SOA lifecycle. With reference to SOA design and development, a relevant work on reusing mainframe assets in SOA based solutions [11] discusses the consideration of mainframe assets. Generic guidelines for assessing mainframe assets in the context of SOA development are also described in [11].

The said approaches primarily explore the repositories space and the generic technical capabilities of interacting with the repositories. They rely on high level of manual involvement in reuse of existing assets that remain disconnected from the tool centric service modeling activities. However, in our paper, we propose an efficient and interactive framework for discovering and selecting different types of assets, and providing a modeling scope to analyze and study the assets in the context of the required solution. We also discuss how our framework with its plug-in based architecture can be extended or customized with respect to supporting new repositories or to cope with changing standards or specifications. In traditional approaches, the candidate assets both at the process and the individual task levels are identified through manual keyword centric search with limited meta data information available. Then, the assets are subjected to fitgap analysis by the architects. After passing through this crucial test, legacy transformation of such assets into actual services as discussed in [8], with or without customization, is considered. Our asset analysis algorithm performs such an optimization that captures dependencies across the business functions (tasks in business processes) that need to be realized either independently or collectively with one or more available assets, and satisfies the specified constraints both at the process as well as at the individual task (business function) levels. Furthermore, our asset analysis algorithm performs multi-level (Base search and Filtered Search) identification for facilitating optimal asset reuse across mutliple repositories.

8 Concluding Remarks

In this paper, we have presented an architectural asset analysis framework and methodology for model-driven development of SOA solutions, involving a variety of business, service, and design models. We have evaluated the asset analysis methodology using the framework during the design of an SOA solution with multiple asset repositories. We have also shown how our methodology supports asset searching based on the solution context and from multiple repositories, thereby improving asset search effectiveness. In future, we plan to work on automated comparison of assets retrieved in a given search, that leads to establishing relationships of similar candidate assets from the same or different repositories. We also plan to establish a multi-dimensional asset modeling and generation framework that helps document asset requirements and constraints as well as capabilities and analytical details of existing assets. This would enable better governance of asset publishing in repositories.

References

1. Arsanjani, A., Allam, A.: Service-oriented modeling and architecture for realization of an soa. In: IEEE SCC (2006)
2. Schmidt, D.C.: Model-driven engineering. IEEE Computer (February 2006)
3. Johnson, S.K., Brown, A.W.: A Model-Driven Development Approach to Creating Service-Oriented Solutions. In: Dan, A., Lamersdorf, W. (eds.) ICSOC 2006. LNCS, vol. 4294, pp. 624–636. Springer, Heidelberg (2006)

4. Arsanjani, A.: Soma-me: A platform for the model-driven design of soa solutions. IBM Systems Journal 47(3), 397–414 (2008)
5. Aguilar-Savén, R.S.: Business process modelling: Review and framework. International Journal of Production Economics, 129–149 (2004)
6. 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)
7. Ponnalagu, K., Narendra, N.C.: Discovering and Deriving Service Variants from Business Process Specifications. In: Bouguettaya, A., Krueger, I., Margaria, T. (eds.) ICSOC 2008. LNCS, vol. 5364, pp. 691–707. Springer, Heidelberg (2008)
8. Zhou, N., Zhan, L.J., Chee, Y.M., Chenr, L.: Legacy asset analysis and integration in model-driven soa solution. In: IEEE SCC (1), pp. 554–561 (2010)
9. Gimnich, R.: Using existing software assets in soa design. In: CSMR, pp. 309–310 (2009)
10. Zimmermann, O., Koehler, J., Leymann, F.: The role of architectural decisions in model-driven soa construction. In: OOPSLA (2006)
11. Chordes, M.: Unleash the power of mainframe assets into soa (2007), ftp://ftp.software.ibm.com/software/rational/web/whitepapers/SWW12599-USEN-00_unleashing_soa_wp_0919.pdf

Reasoning-Based Context-Aware Workflow Management in Wireless Sensor Network*

Endong Tong[1], Wenjia Niu[1], Hui Tang[1], Gang Li[2], and Zhijun Zhao[1]

[1] Institute of Acoustics, Chinese Academy of Science
High Performance Network Laboratory
Beijing, China, 100190
{Tonged,niuwj,tangh,zhaozhj}@hpnl.ac.cn
[2] School of Information Technology
Deakin University, 221 Burwood Highway
Vic 3125, Australia
gang.li@dekin.edu.au

Abstract. Workflow technology is regarded as the automation of an execution process in which the information or tasks are passed from one service to another, according to the predefined execution sequence. Recently, workflow technology has been successfully used in wireless sensor networks (*WSNs*) for service composition. Although workflows can dynamically change according to current context, there is still limited work to facilitate the atomic services reuse.

In this paper, we propose a reasoning-based context-aware workflow management(*Recow*) approach, in which a rule-based reasoning module is responsible for extract semantic information so that the lower sensor data will have a loose couple connection with the upper logic process. By using the semantic information as service I/O, we reconstruct atomic services, then they can be reused in workflow construction. Usually more than one rule will be matched and they can not be executed simultaneously in the rule matching process, so a conflict resolution algorithm is further proposed based on context-aware priority. Finally, two case studies demonstrate that our approach can effectively facilitate resource reuse and also indicate the performance of the *Recow* approach as well as the precision of the conflict resolution algorithm.

Keywords: Wireless Sensor Network, Workflow, Context-aware, Rule-based Reasoning, Resource Reuse.

1 Introduction

Composed of a large number of sensor nodes, wireless sensor networks (*WSNs*) are attractive for physical information gathering in large-scale data rich

* This research is supported by the National Natural Science Foundation of China (No. 60775035, 60802066, 61103158), the National S&T Major Project (No. 2009ZX03004-001, 2010ZX03004-002-01), the National Basic Research Program of China (No. 2007CB311004) the Deakin CRGS 2011 and the Securing CyberSpaces Research Cluster of Deakin University.

G. Pallis et al. (Eds.): ICSOC 2011, LNCS 7221, pp. 270–282, 2012.

environments. *WSNs* also add value to various applications such as surveillance [1], smart home [2] and precision agriculture [3].

However, in order to fully exploit sensor networks for such applications, energy-efficient and scalable solutions for *WSN* services are essential. To address this issue, recent work in *Web Services* [4] [5] has started to utilize the service-oriented architecture (*SOA*) to support resource-constrained *WSN* services by proposing novel protocol stacks. Accordingly, *WSN services* refer to those *Web Services* built on the data gathered from wireless sensors.

One essential characteristic of *SOA* is that it can break a big service into small atomic services which could be distributed over a network and composed into new services. For *WSN* services, *SOA* can bring the benefits of modularity, flexibility, loose-coupling and interoperability. As the key issue of *SOA*, service composition has attracted considerable research interest. Among them, *workflow technology*, known for its practicability and efficiency, has been largely used for service automation and management in industry [6].

Recently, there has been a trend in utilizing *context information* such as environment information into workflows, which forms the research area named *context-aware workflows* [7] [8]. Unlike traditional workflows which only serve static processes, context-aware workflow is more flexible and suitable for dynamic processes on a dynamic scenario.

However, for *WSN* services, the flexibility of context-aware workflow is usually hindered by two problems:

- First, there are a large number of sensors in *WSN*, so we should construct and manage large amounts of atomic services. Also, context frequently updates will further require the workflows to be adjusted accordingly.
- Second but just as important, *WSN* has resources limitations on energy, memory space, computation capabilities and so on. Due to the first problem, it will be a heavy process to construct and dynamically change workflows.

Due to these challenges, traditional context-aware workflows are impractical for *WSN*. Their mechanism doesn't facilitate the atomic service reuse and sharing, and it also frequently make decisions on which concrete workflow should be used to realize the specific task according to current context. Therefore, a mechanism which can limit the number of atomic service and construct much more flexible workflows is indispensable.

If we consider the example of a *temperature control* service where a temperature value is used to determine the status of air conditioner. In different environments, even though service conditions (such as $T > 58$℃, or $T' > 100$℃) are different, they have the same semantic information, namely, *the temperature is high*. In other words, the low-level information (such as the specific temperature value) can be abstracted while the upper-level semantic information (such as $< temperature, is, high >$) can be preserved. If we take this semantic information as input and build a new atomic service in place of existing atomic services whose inputs are concrete temperature data, the number of atomic service will be limited. We depart the reasoning from atomic services through a reasoning model so that atomic services can be reused in many temperature control

workflows due to the fact that the newly built service is loose coupling with sensor data.

The rest of this paper is organized as follows. In Section 2, we present a simple review of related works. The Reasoning-based Context-aware Workflow (*Recow*) management approach is proposed in Section 3, followed by a case study which shows the effectiveness of the proposed approach in Section 4. Finally, Section 5 concludes this paper.

2 Related Work

Since the birth of ubiquitous computing [9] in the 1990s, researchers have attempted to integrate context-awareness into traditional workflows. A variety of context-augmented workflows have been proposed. Wieland et al. [8] proposed a three layered system model which consists of a smart workflow layer (*SWL*), a context provisioning layer (*CPL*) and a context integration layer (*CIL*). The approach proposed by them means that when contexts change, the on-going workflow will be terminated and substituted by a new one.

However, for an environment where the context is frequently getting updated, the efficiency will be critical. In order to address this issue, Ardissano et al. [10] proposed the abstract workflow for context-aware workflow execution (*CAWE*). They utilized the *context manager service* (*CtxMgr WS*) to provide the contextual information and *context-aware workflow manager* (*CA-WF-Mgr*) to execute an abstract workflow as if it were a standard workflow. By using *CAWE*, each abstract service is associated with multiple concrete implementations corresponding to different contexts. Contexts were used to determine the service implementation at run time.

Choi et al. [11] also proposed a framework, in which *context workflow scenario parser* (*CWparser*) represents contexts, while *DItree Handler* decides which subtree to apply. They used contexts as conditions to form a set of service subtrees. Upon the context changes, the workflow will be dynamically reconstructed by adjusting the subtrees. In addition, other context-aware workflow management systems [12] [13] only consider using abstract workflows to bind the business process (the workflow definition level) and the services (the instance level), while the concrete workflow process is determined during execution.

Although automatic workflow construction can be implemented according to the current context, a majority of existing work only considers dynamic service execution, while the resource reuse among services has been largely overlooked. Hence, in this paper, we focus on the facilitating of service reusing/sharing in the context-aware workflows design.

3 The *Recow* Model

From the example of *temperature control* service, new atomic services characterized by semantic information (*the temperature is high*) can be reused by numbers

of workflows. Therefore, through abstracting semantic information, it is possible to achieve resource reuse/sharing in *WSN* workflows. In user-centric services, process logics will frequently change. This means it is more suitable to adopt rule-based reasoning to abstract semantic information because only the rules need to be updated when the process logics is changing.

In this paper, we propose a *Reasoning-based Context-aware Workflow management* (*Recow*) approach, using the framework as shown in Fig. 1.

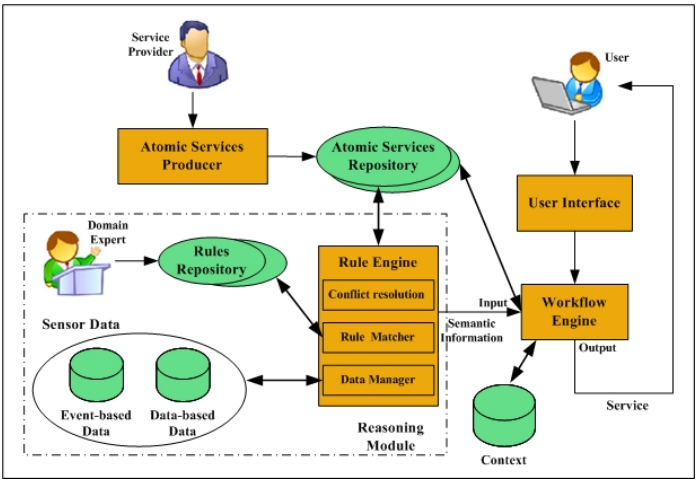


Fig. 1. *Recow* Framework

3.1 The *Recow* Model Overview

As shown in Fig. 1, the *Recow* model consists of three major modules:

- ***atomic services repository*** stores atomic services supplied by service providers for workflow composition.
- ***workflow engine*** which is responsible for building, monitoring and executing a workflow.
- ***reasoning module*** which is used to generate semantic information, as described inside the dotted box in Fig. 1. The captured *Sensor Data* will be used to select appropriate rules from the *Rules Repository* maintained by domain experts. The *Rules engine* is the core of the *reasoning module* and will be discussed further in Sec. 3.2.

The overall working process of *Recow* can be described as follows. Firstly, the service provider designs atomic services whose I/O are characterized with semantic information according to applications. Secondly, domain experts design the reasoning rules through a user-friendly interface. Thirdly, workflows will be

composed from atomic services which stored in the *atomic services repository* to implement working process. In real applications, the low-level sensor data will be imported into the *rule engine* to match designed rules as well as to extract the semantic information. This information is taken as inputs to trigger the workflows.

3.2 Reasoning Module

In *Recow*, we propose *reasoning module*. Firstly, it is possible. We can add the reasoning module between the upper-level business process and lower-level sensor data. Sensor data is initially loaded into the reasoning module, where we get corresponding semantic information. Secondly, it is necessary. Unlike traditional methods, *Recow* build new atomic services whose inputs are no longer low-level sensor data but semantic information. With these atomic services, workflows with improved flexibility and general ability can be composed.

In the reasoning module, the *rule engine* is the core of the reasoning approach, which originates in the expert system [14]. Different from other rule engines in context-aware applications, the rules in the *Recow* model are embedded with semantic information.

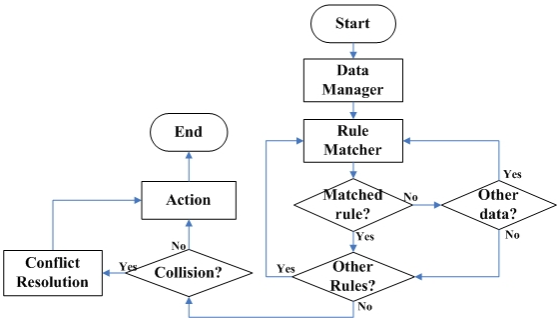


Fig. 2. Flow Chart of Rule-based Reasoning

As shown in Fig. 1, the *rule engine* consists of *data manager*, *rule matcher* and *conflict resolution*. *Data manager* determines which type of sensor data should be loaded into *rule matcher*; *rule matcher* handles the matching between sensor data and rules; *conflict resolution* determines which conflicting rule can be accepted. The rule-based reasoning process is described as a flowchart in Fig. 2. This process contains three components: extended rule representation with context-based priorities, sensor data management and rule matching, and conflict resolution.

Extended Rule Representation with Context-based Priorities. In real world applications, more than one rule could be matched to the same scenario.

Accordingly, a mechanism is needed to resolve the conflicts by deciding which rule will be activated.

The common approach to conflict resolution is based on the priority [15]. When more than one rule is matched at the same time, the rule with the highest priority will be activated. Traditional priority has only one static value. However, in *WSN*, an optimal rule should have different priorities due to different contexts. Accordingly in this paper, the priority part of each rule is extended to accommodate different contexts. Tab. 1 shows a sample rule which decides whether the temperature is comfortable or not.

Table 1. A Rule with Context-based Priority

rule name "temperature"
priority: 5(default),10(indoor),15(outdoor)
object: int temperature.value;
boolean temperature.comfortable;
if:
temperature.value>26(℃) **and**
temperature.value<30(℃);
then:
$temperature.comfortable \Leftarrow true$;
end

In this representation, each rule's *priority* on a specific context is determined by the matched priority. The *object* part of the rule specifies the variables involved in the rule.

In *Recow*, rules mainly come from domain experts. However, in real world, the rule extraction procedure is labor extensive. Therefore, a user-friendly graphical interface to simplify the construction of rules become necessary. For example, the outputs of rules corresponding with the inputs of atomic services, the *Recow* GUI provides drop-down menus for experts to select the sensor data types and semantic information for rules.

Sensor Data Management. Sensor data is usually updated frequently. In this part, we will discuss how sensor data will be controlled and loaded into the *rule engine*.

Rules in the *Recow* model will be constructed and registered into the *rule engine*. At the same time, the data sensed by sensors in *WSN* will be registered into the *data manager*, which is responsible for storing the mapping between the sensor data and the rule variable. For example, for a rule which checks the temperature, the rule variable should be mapped to the temperature data. Based on this mapping, the *data manager* can determine which sensor data will be loaded into the *rule engine*.

For each constructed rule, its rule variables are mapped with some sensor data, which can be roughly divided into two types: the *data-based sensor data* whose value will update frequently, such as the temperature data continuously

gathered by sensors; and the *event-driven sensor data*, whose value updates only when some specific event occurs, such as detection of a glass-break event. Corresponding to these two kinds of data, the *data manager* has two different treatments: the *data manager* will proactively obtain *data-based sensor data* in a fixed time interval, while the *event-driven sensor data* will not be captured until the event-based sensor detects a specific event occurring.

Rule Matching and Conflict Resolution. With respect to rule matching, many rule matching algorithms [16] have been proposed. Among them, *Rete*, an efficient forward inference rule matching algorithm, is one of the most popular matching algorithms.

The *Rete* algorithm starts with constructing a network of nodes, where each path from the root to leaf defines the *condition* part of a rule. When sensor data satisfies the condition of one node, it will propagate down to the next child node through the network until arriving at a leaf. Then the rules corresponding with the path propagated by the sensor data will be matched. Finally, the matched rules are added to the list of candidate firing rules.

In *WSN*, dynamic updating of sensor data will result in repeated rule matching, which costs more in terms of computing resources. We adopt the *Rete* algorithm [17] for its two characteristics. The first characteristic is state preservation, which can avoid duplicated computing on visited nodes. Secondly, data can pass through the network like a flow with a high processing speed. This makes it a suitable choice for large scale rules matching in *WSN* applications.

Nevertheless, conflicting rules remain a challenge when managing a large number of rules in the proposed *Recow* model. For example, let us assume that rule M is "*if air in a home is dry then turn on the humidifier*" and rule N is "*if there is nobody at home then turn off the humidifier*". When the context is "*no one at home and the air is dry*", both rules will be matched. If there is only one humidifier, conflict occurs.

In the *Recow*, we propose a conflict resolution algorithm based on context-based priorities. The pseudocode is provided in Alg. 1, in which *current.context* represents the current context, $Rule_i.action$ represents the *action* part of $Rule_i$ and *priority.context* represents the context with a specified priority in the extended rule.

The basic mechanism of our algorithm is that the rule with the highest priority will be activated. Due to the context-based priorities, the *priority* of each candidate rule will be checked against the current context, and the priority corresponding to the matched context is set as the priority of the rule. Finally, the rules with the highest matched context-aware priority will be activated.

4 Case Study

In this paper, we use two case studies to show the advantages of the *Recow* framework. In the first one, we will illustrate how the proposed *Recow* model

can facilitate the resource reuse and how context-based priority can be used to resolve conflicts. While in the second one, we will give statistical results to show the performance of *Recow*.

Algorithm 1. Context-aware Priority-based Conflict Resolution Algorithm

Require: Conflicted Rule Set $\{Rule_1, Rule_2, ...Rule_n\}$
Ensure: Output the action of selected rule
  $current.priority \Leftarrow 0$;
  $current.action \Leftarrow Rule_1.action$;
  $context\text{-}aware \Leftarrow false$;
  for $i = 1$ to n **do**
    read the rule priority
    if $\exists$ *a priority in* $Rule_i$
    $(priority.context == current.context)$ and $(priority.value > current.priority)$
    then
      $current.priority \Leftarrow priority.value$;
      $current.action \Leftarrow Rule_i.action$;
      $context\text{-}aware \Leftarrow true$;
    end if
    if $context\text{-}aware == false$ **then**
      if $Rule_i.defaultpriority > current.priority$ **then**
        $current.priority \Leftarrow Rule_i.defaultpriority$;
        $current.action \Leftarrow Rule_i.action$;
      end if
    end if
  end for
  return *current.action*

4.1 Process Case Study

A husband (H) and his wife (W) have bought various types of sensors, including infrared sensors, sound detection sensors, etc., to build a smart home system for automatical intrusion detection. There will be various workflows to compose atomic services due to different combinations of sensors.

Let us suppose that a husband and wife build their own workflow in drag-drop way independently as shown in Fig. 3. Workflow H detects the infrared event when there is something around and the sound detection sensor checks whether sound intensity is higher than a threshold. Workflow W detects whether the sound volume is higher than a pre-determined threshold.

Traditional context-aware workflow management systems treat all workflows as concrete ones, from which the working workflow will be selected according to current context. While in the *Recow* model, we can construct a workflow C (as in Fig. 3), which contains a new "*intrusion detection*" atomic service whose input is characterized by semantic information, such as "*intrusion detected*" or "*intrusion not detected*". With the help of the *reasoning module*, semantic information can be extracted. By using the semantic information, such workflow

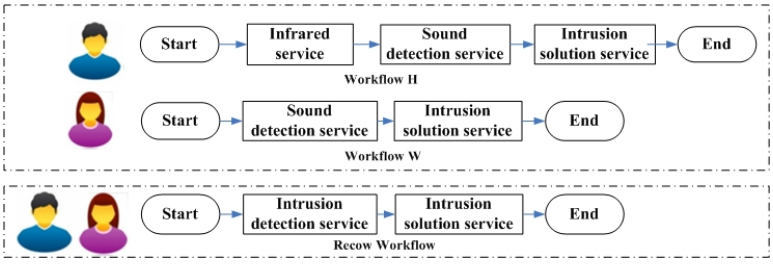


Fig. 3. *Recow* Workflow with Resource Reuse Compared with Original Workflows

can work as an abstract flow which can effectively load both the functions of workflow H and workflow W in the smart home.

In *Recow*, the *infrared service*, *sound detection service* and other similar atomic services can be substituted by the new *intrusion detect service*. Due to the new service's loose coupling with low-level sensor data, when the husband and his wife try to make their own workflow, the *intrusion detect service* can be reused and the same workflow C will be constructed even though different sensor combinations are in use. Therefore the *Recow* model reduces the number of atomic services needed, realizes the resource reuse and improves the reusability of workflow. Furthermore, with context changes, such as different sensor combinations for intrusion detection, workflow C can remain running without dynamically selecting the concrete workflow corresponding to the current context at execution time, which brings more flexibility than traditional context-aware workflows.

In order to get the upper-level semantic information, several rules should be designed initially(See Tab. 2). The default priority of rules depends on the rules' designer. Here, we suppose the wife's rules have higher default priorities than her husband's rules. Suppose the current context in the smart home is: "*infrared device detects some person is around and sound intensity is* 850". This context information will make rule H activated. Then *the reasoning module* can generate an output: "*intrusion happens*", which will be connected to the input of workflow C.

In addition to this, during Chinese New Year celebrations when crackers are released, and the environment is noisy, the system may give a false alarm. Suppose a rule E was constructed to resolve this problem, a new problem emerges: when an infrared sensor detects there is somebody around and sound intensity is 750, rule W and rule E will be matched at the same time but they will be in conflict(rule W outputs intrusion happens while rule E outputs intrusion does not happens). In traditional conflict resolution, only the default priority is used. So rule W will be activated because its default priority is higher, even though it might be a false alarm. In *Recow*, we propose a context-aware priority for conflict resolution. In rule E, the context-aware priority is at 10 during Chinese New Year celebrations, which also means the evening when sound intensity reaches 750, rule E will be activated because of its higher priority. Context-aware priority enables the smart home system to activate the most relevant rules corresponding to the current context.

Table 2. Intrusion Detection Rules

rule name "rule H" **priority**: 6(default) **object**: $(boolean)infrared.status \Leftarrow whethersome\ peopleis\ around\ or\ not$; $(int)(soundintensity) \Leftarrow theenvironment\ sound\ intensity$; **if**: $(infrared.status == true)$ and $((sound\ intensity) > 800)$; **then**: $intrusion.status \Leftarrow true$; **end**
rule name "rule W" **priority**: 7(default) **object**: $(int)(soundintensity) \Leftarrow theenvironment\ sound\ intensity$; **if**: $(sound\ intensity) > 700$; **then**: $intrusion.status \Leftarrow true$; **end**
rule name "rule E" **priority**: 5(default), 10(Chinese New Year) **object**: $(int)(soundintensity) \Leftarrow theenvironment\ sound\ intensity$; **if**: $(infrared.status == true)$ and $((sound\ intensity) < 900)$; **then**: $intrusion.status \Leftarrow false$; **end**

4.2 Performance Case Study

In this paper, a reasoning module is adopted to realize the resource reuse and this brings extra computation cost. In this case study, we will show the performance of the *Recow* approach with the reasoning module. Firstly, we suppose that more people make their rules or one people make more than one rule. So there will be not only above three rules in our intrusion service. We assume that the professional's rules are most suitable when possible conflicts exist. Then, we generate a large number of sensor data to match these rules. Fig. 4 shows the rule matching performance. X-coordinate indicates the number of sensor data used to match rules and Y-coordinate indicates the time consumed in rule matching algorithm.

Furthermore, we test the context-aware priority for the activation of the most suitable rule in *Recow*. We begin by giving several sensor data compositions as shown in Tab. 3. In Tab. 4, we will show the precision comparison of the traditional and context-aware priority used in the conflict resolution algorithm. In Tab. 4, the value "1" and "0" indicates whether the most suitable rule is

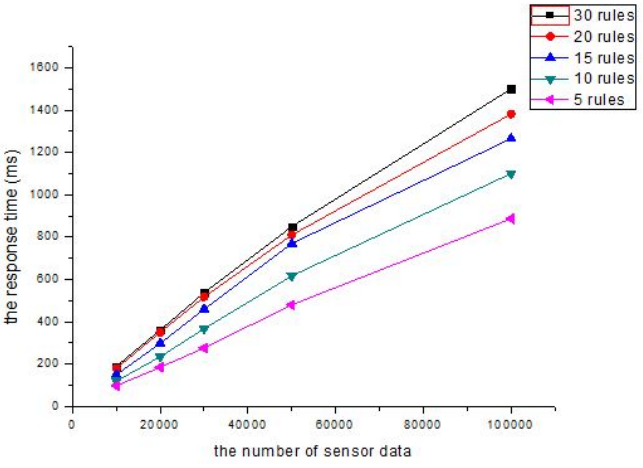


Fig. 4. The Performance of *Recow* Rule Matching

Table 3. The Context List

sensor data composition	description
*composition*1	$infrared.status == true, light == 930, sound\ intensity == 650$
*composition*2	$infrared.status == false, light == 1050, sound\ intensity == 850$
*composition*3	$infrared.status == true, light == 920, sound\ intensity == 750$
*composition*4	$infrared.status == true, light == 700, sound\ intensity == 670$
*composition*5	$infrared.status == true, light == 950, sound\ intensity == 780$

Table 4. The Precision Comparison of Traditional and Context-aware Priority Used in conflict resolution algorithm

	traditional	context-aware	current context
*composition*1	1	1	*Chinese New Year*
*composition*2	1	1	
*composition*3	0	1	*Chinese New Year*
*composition*4	1	1	
*composition*5	0	1	*Chinese New Year*
precision(%)	60%	100%	

activated or not. In this case study, we can see that the time efficiency of the reasoning module does not increase exponentially with the number of sensor data, and the precision of selecting the most suitable rules is also improved by using the proposed context-based priorities.

5 Conclusions

Workflow has been successfully used in wireless sensor networks (*WSNs*) for service composition. Recently, there has been a trend in utilizing context information into workflows. Now, research efforts on context-aware workflows have succeeded to realize the change in workflow according to their current context. However, in resource-limited *WSNs*, we also need to consider resource reuse.

In this paper to enable service resource reuse in context-aware workflows, we propose a reasoning-based context-aware workflow management (*Recow*) approach, in which the rule-based reasoning is exploited. We also present a corresponding resource conflict resolution algorithm. Compared with previous work, our main contributions can be summarized as follows: 1). Unlike traditional context-aware workflows, *Recow* can build new atomic services in place of existing ones. We characterize the new atomic service I/O with semantic information generated by a reasoning module, which effectively separates process logic and underlying sensor data. Hence, atomic services can be reused and furthermore, *Recow* can support context-aware workflow better. 2). We propose a context-based priority in the resource conflict resolution algorithm, through which we can select the most suitable rule for the current context.

The case studies presented in this paper demonstrate that *Recow* can effectively facilitate resource reuse and the extra computation time consumed in rule matching is acceptable. They also demonstrate the precision of the conflict resolution algorithm based on context-aware priority.

References

1. Wang, X., Wang, S., Bi, D.: Distributed visual-target-surveillance system in wireless sensor networks. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 39(5), 1134–1146 (2009)
2. Suh, C., Ko, Y.B., C, C.: Collaborative workflow solution for distributed product development. In: 12th International Conference on Computer Supported Cooperative Work in Design, pp. 594–599. IEEE Press, Xi'an (2008)
3. de Lima, G.H.E.L., Neto, P.F.R.: WSN as a Tool for Supporting Agriculture in the Precision Irrigation. In: 6th International Conference on Networking and Services, pp. 137–142. IEEE Press, Cancun (2010)
4. Delicato, F.C., Pires, P.F., Pirmez, L., da Costa Carmo, L.F.R.: A Flexible Middleware System for Wireless Sensor Networks. In: Endler, M., Schmidt, D.C. (eds.) Middleware 2003. LNCS, vol. 2672, pp. 474–492. Springer, Heidelberg (2003)
5. Leguay, J., Lopez-Ramos, M., Jean-Marie, K., Conan, V.: An efficient service oriented architecture for heterogeneous and dynamic wireless sensor networks. In: 33rd IEEE Conference on Local Computer Networks, pp. 740–747. IEEE Press, Montreal (2008)
6. Tsai, M.J.: The workflow development for electronic manufacturing industry. In: 12th International Conference on Computer Supported Cooperative Work in Design, pp. 688–692. IEEE Press, Xi'an (2008)
7. Cho, Y., Choi, J., Choi, J.: A Context-Aware Workflow System for a Smart Home. In: 10th International Conference on Computer and Information Technology, pp. 95–100. IEEE Press, Dhaka (2007)

8. Wieland, M., Kaczmarczyk, P., Nicklas, D.: Context integration for smart workflows. In: 6th IEEE International Conference on Pervasive Computing and Communications, pp. 239–242. IEEE Press, Hong Kong (2008)
9. Weiser, M., C, C.: Ubiquitous Computing. Computer 26, 71–72 (2008)
10. Ardissono, L., Furnari, R., Goy, A., Petrone, G., Segnan, M.: A framework for the management of context-aware workflow systems. In: 3rd International Conference on Web Information Systems and Technologies, pp. 1–9. IEEE Press, Florian Lautenbacher (2007)
11. Choi, J., Cho, Y., Choi, J.: A Context-aware Workflow System Supporting a User's Dynamic Service Demands in Ubiquitous Environments. In: 1st International Conference on Multimedia and Ubiquitous Engineering, pp. 425–430. IEEE Press, Seoul (2007)
12. Chaari, T., Ejigu, D., Laforest, F., Scuturici, V.M.: A comprehensive approach to model and use context for adapting applications in pervasive environments. Journal of Systems and Software 80(12), 1973–1992 (2007)
13. Modafferi, S., Benatallah, B., Casati, F., Pernici, B.: A methodology for designing and managing context-aware workflows. Mobile Information Systems II, 91–106 (2005)
14. Jackson, P.: Introduction to expert systems. Addison-Wesley Longman Publishing Co., Inc., Boston (1998)
15. Thyagaraju, G.S., Math, M.M., Kulkarni, U.P., Yardi, A.R.: Conflict Resolving Algorithms to Resolve Conflict in Multi-user Context-Aware Environments. In: International Conference on Advance Computing Conference, pp. 202–208. IEEE Press, Patiala (2009)
16. Karp, R.M., Rabin, M.O.: Efficient randomized pattern-matching algorithms. IBM Journal of Research and Development 31(2), 249–260 (2010)
17. Xiao, D., Zhong, X.: Improving Rete algorithm to enhance performance of rule engine systems. In: 2nd International Conference on Computer Design and Applications, pp. 572–575. IEEE Press, Qinhuangdao (2010)

Service for Crowd-Driven Gathering of Non-Discoverable Knowledge

Jim Laredo[1], Maja Vukovic[1], and Sriram Rajagopal[2]

[1] IBM T.J. Watson Research Center, Hawthorne, NY 10128, USA
[2] IBM India Pvt. Ltd., Chennai - 600 089, India
{laredoj,maja}@us.ibm.com, srirraja@in.ibm.com

Abstract. Knowledge that cannot be discovered through automated methods, such as user practices, remains in informal mediums. It is unstructured, and in collective possession of the experts, yet it is key for business insights. Typically this "Non-Discoverable knowledge" is gathered in semi-automated way, which at best provides crude estimates, and doesn't scale. In this paper, we describe our novel approach to rapidly design a process solution for a family of business objects, gathering required knowledge through the use of social networking to identify the experts. We propose a "Deconstructed Survey" that captures the knowledge request, and manages its lifecycle through task forwarding and sub-tasking. We developed the system BizRay, instantiating the proposed approach as a general-purpose, self-service Web-based, crowdsourcing service. We demonstrate its effectiveness in accelerating knowledge discovery, through our experiences with deployments for IT Optimization and Services Delivery.

Keywords: crowdsourcing service, knowledge discovery, people intensive, business network, social network, exception handling, rapid design.

1 Introduction

The biggest strength of an enterprise is its people, specialists who understand and drive the business and the IT. What is the best way to tap into this knowledge and organize powerful knowledge networks to help ease and accelerate important initiatives within an organization? If experts are recruited beforehand as when writing a book, it is possible to subdivide the writing and assign sub-topics to each one. In this case it is possible to identify the people beforehand and subdivide the task at hand. A different type of knowledge capture is required when a business process has taken place, or time has passed and the subject has evolved through time. Experts that contributed to the business activity gained some knowledge that may or may not have recorded it for others to learn. Systems involved in the process management likely have recorded, through logs, the events that took place, even captured the interactions people may have had with the systems. This information is readily available and many, such as Li, *et.al.*, [1], van der Aalst, *et.al.*,[2], Neumann and Robeller [3] have explained techniques to process these logs for the sake of process

G. Pallis et al. (Eds.): ICSOC 2011, LNCS 7221, pp. 283–294, 2012.

understanding, troubleshooting or compliance auditing. For those portions of the process where a system was not involved, such as the process to reach a decision, or the very knowledge to use or access the system, the knowledge most of the time remains within the participants and is never extracted. Some unstructured forms like documentation, spreadsheets, or even wikis may capture some of this information but usually requires a proactive approach to store it and to retrieve it. We call "non-discoverable information" the one that the system cannot capture, lacks consistent structure and remains with those individuals who were involved in the process.

Non-Discoverable information is pervasive in domains such as IT Services Management, and traditional communications methods have been used, such as emails containing pre-formatted forms or spreadsheets, to capture it. A small team initiates emails for each knowledge request to be fulfilled, waits for responses, captures replies, assesses completeness and follows up if needed. In addition a great deal of time and corresponding resources needs to be dedicated to exception handling. The most common exceptions are: 1) *Non-response*: they need to be followed up by reminders or management escalations; 2) *Incorrect target:* when the expert has moved on to another role and has forgotten or has no access to current information; 3) *Unavailability:* when targeted expert may be away or no longer works in the organization. 4) *Requests for more information:* in this case the recipient of the request is not clear on how to proceed and usually replies back with questions. Traditional approaches can work when the number of initial requests is small but have difficulty scaling to larger numbers. In our initial study [4], we employed crowdsourcing to engage experts to capture compliance and application-hosting specifications. Previously a small team of 2 people was able to handle 140 requests in a 2-month period, which was acceptable. They created a spreadsheet and through email notified potential stakeholders that could source the information, they correlated emails and acted based on the responses received. However, at that pace, for the target of 4500 collections it would have required more than 64 months to complete. Instead our approach for the overall target took 10 weeks, with close to 90% completion achieved in just 5 weeks, and 50% completion in only 4 days.

In this paper, we describe how we extended a custom web application to a general-purpose, Web-based, enterprise crowdsourcing self-service expediting design and delivery of crowdsourcing campaigns. To achieve that we have designed a concept of a deconstructed survey, enabling request delegation and sub-tasking.

The key contributions of our proposed approach and its realization as a crowdsourcing service, called BizRay, are:

1. Model for deconstructed survey: Consists of the underlying artifacts for defining a survey and capturing and managing for each instance the responses and flows in the form of tasks, similar to a worklist in a workflow system [5]. The model facilitates delegation of requests and their subtasking to others. It generalizes our approach and accelerates deployment of crowdsourcing campaigns.
2. Centralized capture of the knowledge: the system implementation provides a central repository that stores and organizes all the responses, which eliminates

labor-intensive email correlation and transcription of the results received via e-mail and instant messaging.

3. Social network features: More than one user can complete each task. If the information gathered is partial or unknown, the user has the choice to forward the task to another party of their choosing and request their assistance. As people contribute their knowledge, the system keeps track of their identity resulting in the formation of micro communities around the object of that inquiry.
4. General purpose and self-service: Generalization and self-service enablement has significantly shortened the design time to be able to accommodate the different crowdsourcing campaigns, often in less than a day; including interoperability with other systems.

The paper is structured as follows. Section 2 reviews related work in crowdsourcing. Section 3 provides an overview of the proposed approach. Section 4 describes data model, architecture and implementation specifications of the BizRay service that realizes the proposed model. Section 5 presents our experiences from the deployment of the proposed model. Finally Section 6 concludes and outlines future work.

2 Related Work

Early examples of crowd engagement can be found in the marketing domain, where companies have run competitions for end-users to contribute towards certain enterprise functions, such as advertising campaign design or specific problem resolution challenges. Xerox's Eureka system [6] is an example of knowledge harvesting system within the scope of the enterprise. ReferralWeb [7] allows for content co-creation, enabling end-users to become aware of their existing communities that are generated by data mining of Web documents and forums. Wikipedia [8] relies on motivating humans to share information and by that either gain creditability or obtain the equivalent information. Amazon's Mechanical Turk[9] is a platform for micro-tasks (e.g. image tagging, classification and transcription).

SurveyMonkey [10] and AdobeFormsCentral[11] are existing commercial solutions, which enable Web-hosted, customizable surveys enabling opinion from end-users. They do not provide mechanism for survey delegation and sub-tasking, which are the core contributions of our work. Facebook Poll application allows for surveys within social network, however it is dependent on the existing network, whereas in our approach network is iteratively built as survey is being propagated.

3 Deconstructed Survey: Capturing Non-Discoverable Knowledge

Knowledge gathering is a fragmented process, it requires hopping from source to source and inquiring at each step for the missing elements. It is important to know

who the potential sources might be or at least know someone that may know where to inquire next. If the source is not known, the possibility of crowdsourcing with an open call, as described by Howe [12] is an option. Depending on the setting crowdsourcing may or may not appeal the crowd that may be targeted. For example when looking for information on a topic, the setting of a forum with people that share interest in that topic may prove successful. The open call replaces the need to search for the expert, and anyone available may answer the inquiry. However, when seeking information on a specific object, the potential community that may respond successfully is much smaller and may not be necessarily available for such inquiries. In this case it is easier to establish a chain or several chains of inquiries until the necessary knowledge is captured. As a result two challenges surface: 1) how to identify the next contributor(s); and 2) how to collect fragmented knowledge.

In this section we present our deconstructed survey model and address exceptions listed in Section 1. Figure 1 shows a sample survey, automatically generated from a given template by our BizRay system, described in Section 4. This figure shows a subset of the questions about an asset, in this case business application.

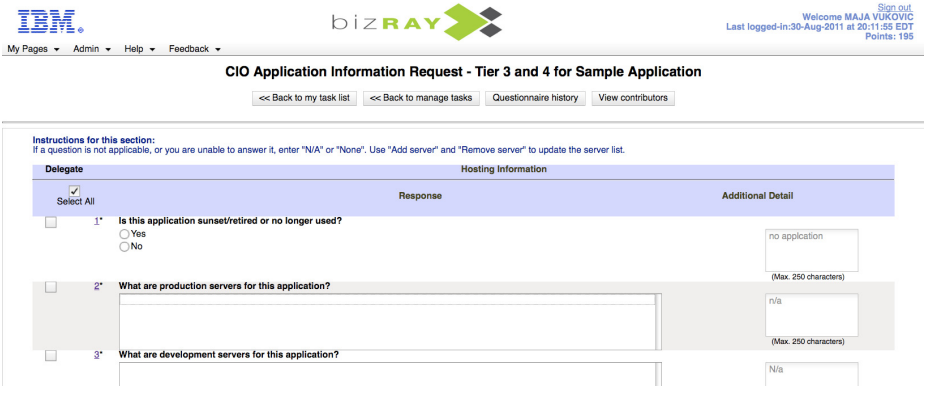


Fig. 1. Sample Survey

There are two main data artifacts that we handle and for which we define a lifecycle: 1) the survey, or knowledge that we want to capture and 2) the task that can be associated with a survey. Note that there is a *0 to n* relationship between the two. For a given survey, there can be 0 or more tasks open for knowledge gathering.

3.1 Surveys

In the first version of our system [4], the knowledge gathering information was custom to the use case, and based on a custom User Interface. The design and implementation iteration proved too lengthy and it became a necessity to have a quick mechanism to facilitate the definition of the use case. A form based approach, allowing for quick design was implemented. We created typical data collection fields.

The notion of a survey was a natural representation of the knowledge capture, it allows for quick design, and allows structuring the data elements that are being gathered. It is likely that a question may not be addressed as specified, as answers received may not be part of the original choices. For that, an optional comment field allows for the "write-in" opinions in those questions that disambiguation is required.

Our surveys are not about someone's opinion but rather the knowledge around a subject or object, a topic or an asset. We do not expect one person to be the sole contributor but rather a team of specialists. A fraction of a survey is a survey, that is, it allows us to break down a survey in parts and treat the resulting parts as surveys on their own, this creates a natural recursion to break down and reconstruct the parts of a survey as we request and address contributions from other parties. A survey artifact is mapped directly to the user interface perspective; it is divided in sections and for each section multiple questions are possible. A question is of a given type, based on the type of an answer expected, and represented in the user interface as described above. An *isMandatory* attribute determines if a question is mandatory or optional. As the name of the attribute indicates, only mandatory questions need to be completed to close a survey.

It is possible that there may be more than one answer for a given question. It is important to timestamp and track the person that provided each answer. This history will present the chronology of answers, with the latest response being the current one. More importantly the history trail forms a small community around each question, subsection or the overall survey. At the end of the survey, a group of people that are knowledgeable about the subject that was being investigated has been captured. This micro-community could be applied to either validate any data entry or perhaps gather further details on the subject in question or a related subject.

3.2 Tasks

A task is a unit of work associated with a portion of the survey. When a survey is launched, the top task is created. The user has several choices to manage the task:

1. Complete the survey. Answer all mandatory questions to the best of his ability.
2. Forward the task to another expert. As a result the responsibility of the survey is transferred to someone else.
3. Invite others. Segment the survey and invite other experts to contribute to the selected questions. A new task is created for each invitation generated, the current parent task remains open until all the mandatory questions associated with the survey are addressed.

Once user accepts the task he has the same set of choices:

1. Complete the questions associated with that task. If all mandatory questions are completed the task is closed.
2. Forward the task to a more knowledgeable party. The responsibility of the tasks is transferred to someone else.

3. Invite others. Segment the task and invite other parties to contribute to the corresponding sections. A new child task is created for each invitation generated, the current parent task remains open until all the mandatory questions associated with the parent task are addressed.

In essence a task tree is created. Figure 2 shows an example, where T0 is the task associated with the whole Survey and subsequent tasks capture a subset of the parent task questions, for example, T4 will include a number of questions that are also part of T2 that belong to the survey that T0 is managing.

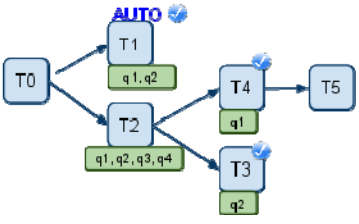


Fig. 2. Task Completion Sequence

Accepting Tasks

There are two basic modes of accepting a task, by assignment and by selection. By assignment occurs when the task is pre-assigned by a system or another person. By selection occurs when a person chooses a task that he is able to manage and proceeds to complete according to the steps described above. In the former case the notion of a network (business or social) is leveraged to identify potential users that can assist in completing the task. In the latter case, the task is available as an open call ready to be crowdsourced, anyone willing or meeting required criteria may accept the task and proceed to handle it. Whoever accepts a task we call the Task Owner. We introduce the notion of rejecting a task to allow for those cases where the assignment of the task has been performed incorrectly in that case the task is returned to the previous owner, if the task was just created it is assigned to the parent task owner.

Completing Tasks

For a task to be completed, it is sufficient that all its mandatory questions be completed. Since there are no constraints on the questions that form the tasks, other than being a subset of the parent task, a question may find its way to other tasks and as such once completed in one task it will contribute to the completion of those tasks that also have that question. Figure 2 further illustrates the task completion process. Assume that T1 has questions q1 and q2, and T2 has questions q1, q2, q3, and q4, and T3 has questions q2, and T4 has questions q1, all questions mandatory. When T4 completes, it means q1 has been answered. This is not sufficient to complete T1 because q2 is also required for T1 to complete. When T3 completes, it means that q2 has been answered. At this point, T1 will also complete, as both q1 and q2 have been answered. The task will be automatically removed from T1's task list.

There are several reasons to adopt this approach. First, it accelerates the completion of the survey. In essence the asynchronous completion of a task prunes the whole sub tree underneath it, basically all tasks complete by transitivity as they only have a subset of the completed questions. Second, although multiple answers may help validate or identify additional detail, this approach helps identify willing and strong knowledge sources. In the example above T3 and T4 task owners together are likely to be more willing than T1's task owner. This information can be applied by T0 to decide to target additional questions to the willing users. Finally, it allows creating a notion of collaboration, two or more tasks can be created with similar set of questions, and assigned to different owners. Each owner is now contributing to a common effort and as the questions get answered it is possible to share the knowledge and allow shifting focus to those questions that still require attention.

When a task completes, the parent task owner is notified. This allows for the parent task owner to track progress, and allow him to decide if he concurs with the answers provided, and to pay attention to the questions that are still open. For tasks owners for which their task was automatically closed, they will get notified if they had opened already the task, otherwise the task will simply be removed from their task list.

It is possible for administrators to cancel tasks in progress, reopen canceled/ completed tasks and transfer tasks to other users. Administrators can search for tasks based on the user it is assigned to, date of assignment and state of the task. Administrators can close surveys that are no longer required which in turn cancels all pending tasks. Administrative tasks come in handy when handling exceptions that have not been considered.

3.3 Exception Handling

The ability to forward a task and reassign is a necessary mechanism to deal with leaf nodes in the tree that would otherwise end in a dead end. The task may have reached a user by accident by incorrectly typing their name or address. Either the task is rejected and reassigned to the parent, or someone that at least may be familiar with a more knowledgeable party has the chance to forward the request to that party.

When dealing with unresponsive users, but possibly knowledgeable it is necessary to send reminders and escalations, manually and automatically, for example, reminders can be sent for tasks beyond a certain age (in days). Reminder intervals can be set as a function of the age of the task, and level of escalations can be determined based on the number of reminders that have taken place.

The history of responses provided by users for each question is recorded. This provides information about who, when and what information was provided for each question. The history information, which includes details of the user and timestamp, can be viewed at the task level and also at each question level. We have used this information to understand the relationships amongst those that forward or create sub tasks and to whom they engage in their knowledge capture. In some cases this information can point us to other potential contributor, we have studied this problem in part in [13].

4 System Architecture

This section describes the BizRay system, which instantiates the proposed framework for social gathering of distributed knowledge gathering. We describe the service components and how their interactions realize a self-service Web-enabled, software as a service enterprise crowdsourcing service that allows us to expedite design and delivery of crowdsourcing campaigns. We describe the internal representation of knowledge in BizRay and the technologies that were used to implement, deploy and host this system.

4.1 Knowledge Representation

BizRay employs artifact centric approach to modeling business objects and processes upon them. Figure 3 shows the set of data entities that capture survey, its questions, answers and relationship to the user profile. Responses to knowledge requests are stored in the `SurveyResponse` artifact. The `answerChoiceIdList` field stores the list of IDs of selected options in case of multiple-choice questions. The same field is used to store a list of values for questions that allow users to enter more than one value as an answer.

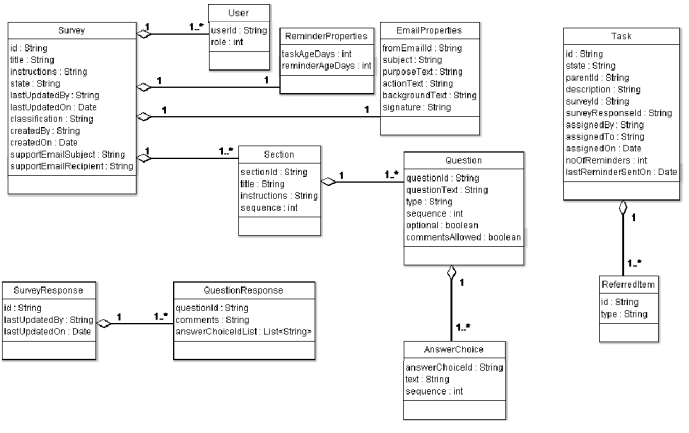


Fig. 3. Class diagram representing survey data model

When a user forwards or segments a task, a new task instance is created which contains the set of questions referred (as a list of ReferredItem instances). In case of a forward, all the questions referred to the forwarding user are part of the new task. For segmentation, the user has to select one or more questions, which form new task instance. For a given survey instance, a user might have more than one task assigned against his/her name. These tasks are consolidated into a set of logical tasks based on the instance id while displaying the task list and for other actions in the system.•

4.2 Services Platform

Figure 4 shows the overview of the BizRay components that implement our approach. At design time, a simple configuration file in the form of a spreadsheet allows to

define the list of questions and sections that are part of a survey, capturing question type, optional comments, and mandatory flag. QuestionnaireManagement service enables survey creation, activation and closure. Once the survey is activated, administrator can assign task instances to the initial pool of users, by uploading a spreadsheet with task assignments. Expert Discovery Service[13] leverages internal social networking services to proactively discover alternate task assignees for tasks that are opened.

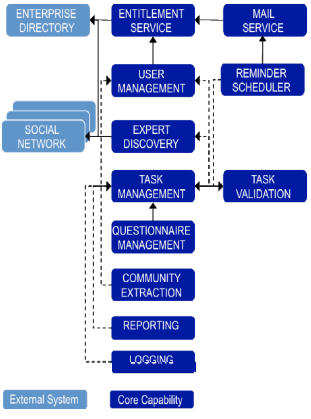


Fig. 4. Components providing the core capabilities of the framework

BizRay is built using Struts to support Model-View-Controller (MVC) design approach. It uses Java Server Pages (JSP), Servlets, HTML, JavaScript and Dojo for rendering the UI. Velocity framework is used to provide template capabilities, in particular to create dynamic email content. BizRay connects to the IBM LDAP server using Enterprise directory API (called BluePages) and to the database using Siena[14] (an artifact centric business process modeling tool developed by IBM tool which enables REST based communication).

5 Experiences in Deploying Deconstructed Surveys

Over the past two years we have been deploying the BizRay system, as the implementation of the deconstructed survey approach, to accelerate knowledge discovery in Services Delivery and IT Optimization domains. Specifically, the crowdsourcing campaigns have often been run to discover knowledge about a particular asset. E.g. in the IT Asset Management domain, by asset we mean a system, server or a business application that is hosted in the infrastructure. However, in the context of the BizRay system itself, an asset can be any business or IT entity whose properties you want to discover and enrich.

Figure 5 shows a deployment setup of 4 new use cases, in terms of number of tasks triggered, completed, complexity of the survey (number of questions) and number of reminders triggered.

USE CASE	DPP	T1T2	T3T4	CostSupport
Number of tasks created	2334	2047	6001	1095
Number of tasks completed	1102	1787	4054	842
Number of tasks forwarded	522	1370	2740	569
% Tasks completed	47.22	87.30	67.56	76.89
% Tasks completed on 1st day	20.24	17.53	18.51	11.25
Day of the first reminder	22	21	91	7
% Tasks completed before 1st reminder	34.53	43.52	46.04	20.27
Total number of reminders	1	5	4	4
Number of questions / task complexity	5	8	8	17
Duration of survey (in days)	40	120	120	120

Fig. 5. Deployment setup of surveys

In T1T2 and T3T4 use cases we are collecting hosting and compliance data for 7000 applications in order to ensure solutions must conform to the regulation(s), requiring that the applications be hosted within a particular country and administered only by persons that are legal citizens of that country. T1T2 and T3T4 denote tier (value/importance classification of the application). For a portion of this use case, 1800 applications were selected, 12% of the tasks opened where child tasks, similar to the results we saw in [4], and close to 50% of the tasks were reassigned to a better source, and within 8 weeks 60% completion was achieved. BizRay was further deployed to capture insights about defect prevention process (DPP) in the services delivery domain. This 5-question survey was assigned to 2334 quality analysts to better understand the impact of tools used for defect prevention. This case is a good collaboration example – over 25% tasks were forwarded. In CostSupport, Survey campaigns were centered about discovering financial and support application data, 1000s of applications, in the context of Application Portfolio Management, to eliminate significant fraction of current IT expenses needed to be evaluated. This particular use case leveraged use of BizRay system to eliminate the effort of data transcription (when surveys are sent out and managed through e-mail), as well as to support knowledge request size scaling (transcribing 4 questions per survey is somewhat "manageable", as opposed to having to manually collect and transcribe e.g. 17 questions, as was the case in this scenario).

Figure 6 shows how the task completion evolved in three different surveys: T1T2, T3T4, and DPP. Figure 7 shows the effect of reminders on T1T2 survey, and visualizes how reminders have a blasting effect on task completion. In all of these use cases, we have seen exploding patterns that coincide with the notification of tasks open. Simpler tasks, those that inquire up to 10 data elements are attended to, much quicker than complex tasks (e.g. that may require lookup to server repositories, etc.). In some cases complex tasks those that would take more than a few minutes are often left unattended and require several escalations.

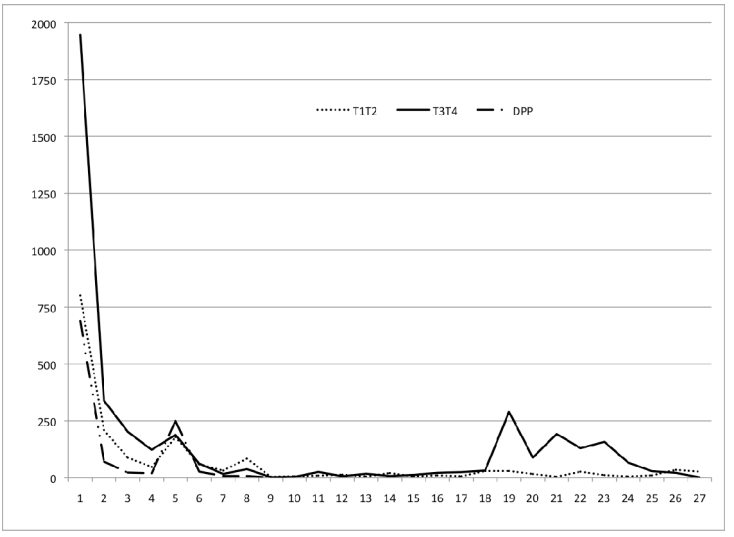


Fig. 6. Number of tasks completed in 5-day segments

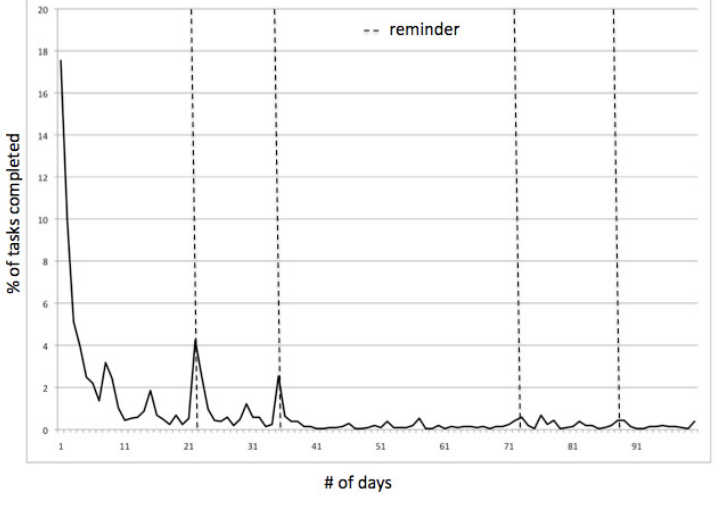


Fig. 7. Effect of reminders on task completion

Use case design has been shortened to about a person-week, often including the tool introduction, requirement gathering, and initial target community identification; and deployment in a multi-tenant environment is down one day. Finally, as many of these use cases may trigger information requests about the same application, BizRay is becoming a central point for managing the data about current applications, servers, etc. Updated data on e.g. application owners is fed into subsequent surveys.

6 Conclusions and Future Work

We extended our approach by designing a generic form to describe a survey structure and a task lifecycle. The deconstructed survey can be broken down into smaller pieces and distributed to multiple users. The approach is flexible enough allowing to be applied to a variety of use cases that seek knowledge discovery. This generic process has proven useful when scaling to a large number of instances, such as assets in a data center. Our ongoing and future work falls into the following areas:

1) Extend the knowledge capture process to context-based follow-ups.
2) Design incentives suitable for enterprise environments, building on the work in the behavioral economics [14].
3) Reuse communities that are discovered in one use case, to uncover sources of information for subsequent ones.

References

1. Li, N., Kang, J., Lv, W.: A hybrid approach for dynamic business process mining based on reconfigurable nets and event type. In: ICEBE 2005 (2005)
2. van der Aalst, W., Weijters, T., Maruster, L.: Workflow mining: discovering process models from event logs. IEEE Transactions on Knowledge and Data Engineering 16(9), 1128–1142 (2004)
3. Neumann, F., Robeller, A.: Enhanced audit log data analysis and query for BPEL processes with Process Choreographer 5.: IBM DeveloperWorks (2005), http://www.ibm.com/developerworks/websphere/library/techarticles/0503_neumann/0503_neumann.html
4. Vukovic, M., Lopez, M., Laredo, J.: PeopleCloud for the Globally Integrated Enterprise. In: Dan, A., Gittler, F., Toumani, F. (eds.) ICSOC/ServiceWave 2009. LNCS, vol. 6275, pp. 109–114. Springer, Heidelberg (2010)
5. Workflow Management Coalition, The Workflow Reference Model (1995)
6. Bobrow, D.G., Whalen, J.: Community Knowledge Sharing in Practice: The Eureka Story. Journal of the Society of Organizational Learning 4(2) (Winter 2002)
7. Kautz, H., Selman, B., Shah, M.: Referral Web: Combining Social Networks And Collaborative Filtering. Communications of ACM 40(3), 63–65 (1997)
8. Bryant, S.L., Forte, A., Bruckman, A.: Becoming Wikipedian: transformation of participation in a collaborative online encyclopedia. In: GROUP 2005: Proceedings of the 2005 International ACM SIGGROUP (2005)
9. Amazon Mechanical Turk, http://www.mturk.com
10. Survey Monkey, http://www.SurveymMonkey.Com/
11. Adobe FormsCentral, https://formscentral.acrobat.com/welcome.html
12. Howe, J.: The Rise Of Crowdsourcing. Wired 14(6) (June 2006)
13. Ypodimatopoulos, P., Vukovic, M., Laredo, J., Rajagopal, S.: Server Hunt: Using Enterprise Social Networks for Knowledge Discovery in IT Inventory Management. In: SERVICES (2010)
14. Cohn, D., Dhoolia, P., Heath III, F., Pinel, F., Vergo, J.: Siena: From PowerPoint to Web App in 5 Minutes. In: Bouguettaya, A., Krueger, I., Margaria, T. (eds.) ICSOC 2008. LNCS, vol. 5364, pp. 722–723. Springer, Heidelberg (2008)
15. Ariely, D.: Upside of Irrationality. Harper (2010)

Improving Service Processes with the Crowds

Donghui Lin[1], Toru Ishida[2], Yohei Murakami[1], and Masahiro Tanaka[1]

[1] National Institute of Information and Communications Technology (NICT),
3-5 Hikaridai, Seika-Cho, Soraku-Gun, Kyoto, 619-0289, Japan
[2] Department of Social Informatics, Kyoto University,
Yoshida-Honmachi, Sakyo-Ku, Kyoto, 606-8501, Japan
{lindh,yohei,mtnk}@nict.go.jp, ishida@i.kyoto-u.ac.jp

Abstract. With the development of service-oriented computing, more and more Web services are provided for users. However, there are situations that services or service processes cannot meet users' requirements in functional QoS dimensions (e.g., translation quality in a machine translation service). Meanwhile, the emergence of crowdsourcing makes various types of tasks done in more and more efficiency ways with low costs. To consider both functional QoS and non-functional QoS, in this paper we try to combine crowdsourcing activities with service processes. Further, this study aims at analyzing the effects of crowd activities on service processes in the real world. We use a case study in the domain of language service with a large scale experiment to show that composing crowd activities and Web services brings variety and creativity to the traditional service processes and human processes. From the experiments and analysis, we find out that quality of crowd activities is essential to service processes, and that crowd activities with high quality can significantly improve various QoS dimensions of the service processes.

1 Introduction

The power of crowds is bringing more and more opportunities for modern business. The emergence of crowdsourcing makes various types of tasks done in more and more efficiency ways with low costs. It is said that there is almost no limitation of crowd resources in crowdsourcing. There were reportedly more than 100,000 workers in over 100 countries (year 2007) in Amazon Mechanical Turk. The mobile crowdsourcing site Txteagle has 2.1 billion mobile phone subscribers across almost 100 developing countries. However, most crowdsourcing businesses are described as "humans doing human work" because people are just doing simple work with low salaries there. In service oriented platforms, it becomes necessary to compose crowd activities with services to bring creativity to the traditional service-based business processes.

In QoS-aware service composition, general QoS dimensions are always defined, including execution cost, execution time, reputation, availability and so on [1], which are very important to evaluate non-functional quality of atomic services and composite services. Besides, many approaches of computing QoS from multiple dimensions have been reported [1,2,3]. However, application-specific QoS

G. Pallis et al. (Eds.): ICSOC 2011, LNCS 7221, pp. 295–306, 2012.

dimensions might also be essential in many cases. In the case of translation services, users care about the translation quality[1] rather than general dimensions. Therefore, it is important to keep the translation quality while considering the improvement of other QoS dimensions.

To address the above issue, combining Web services and human activities is expected to be a promising solution. A good example in language service domain is that translation work can be done by composing machine translation services, monolingual crowd users and bilingual crowd users. As a first step, we conducted a preminalary experiment [5] to show the effectiveness. However, the problem is that human activities might be the bottleneck if human resources are not sufficient or human task quality cannot be guaranteed. Therefore, in this paper we try to use the crowdsourcing approach to improve service processes, considering both functional QoS and non-functional QoS: service-based processes are efficient; human participation guarantees functional QoS; crowdsourcing brings cost reduction.

The rest of the paper is organized as follows: Section 2 provides a motivation example in the language service domain. In Section 3, basic types of composing Web services and crowd activities are defined for QoS improvement. Section 4 and Section 5 introduces a case study in the language service domain with experiments, analysis and discussion. Section 6 introduces some related work. In Section 7, we will conclude the work.

2 Motivation Example

In this section, we show an example of language translation. To provide flexible language services, we developed the Language Grid [6], which has been collecting language resources from the Internet, universities, research labs and companies. All the language services are wrapped from language resources by standard Web service interface. Using the atomic Web services, we have also developed a series of composite services[2]. Figure 1 shows a composite machine translation service which combines services including morphological analysis service, dictionary service, machine translation service and so on. By combining dictionary services and other services, the translation quality can be improved.

However, although many types of services/processes are provided, there still exist limitations in functional QoS dimensions, e.g., machine translation services can never have perfect fluency and adequacy even when they are combined with dictionaries or other services for QoS improvement. That means service-based processes are not able to meet users' requirements in some cases. For example, composite service in Figure 1 might be able to deal with the QoS requirement for online chatting, while it is difficult to use a pure service-based process to write business documents or translate the product operation manuals. To study this problem, we conducted a preminalary trial to combine human activities and

[1] In the language service domain, translations were evaluated on the basis of adequacy and fluency in previous reports [4].

[2] http://langrid.org/service_manager/language-services

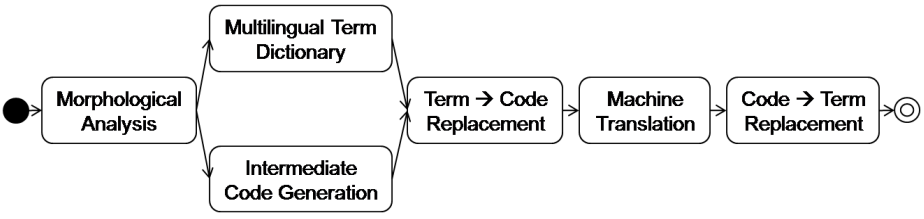


Fig. 1. A composite machine translation service provided by the Language Grid.

Web services [5] with a small scale experiment. However, we found that human might also be the bottleneck if human resources cannot be assured or human task quality cannot be guaranteed. Therefore, we consider combining crowdsourcing activities into service process. Although this example falls into the language service domain, the problem that pure service-based processes cannot always meet users' requirements due to limitations in functional QoS dimensions do widely exist in many other domains.

3 Composition of Web Services and Crowd Activities

Considering that there is an existing service process, crowd activities are possible to be introduced by substituting an atomic service (or a subprocess), forming a selective control relationship with a service (or a subprocess), or processing the input or output of an atomic service (or a subprocess) completely or partially. Moreover, the crowd activities can also be introduced into a process composed by crowd activities and Web services using the same approach. There are following basic types for introducing a crowd activity into a service process for improving certain QoS dimensions $Q^S(s)$.

- *Complete substitution*: a crowd activity ca_i is used to substitute a service s_i (or a subprocess) completely, i.e., $(Q^S(ca_i) > Q^S(s_i)) \rightarrow S(s_i, ca_i)$ ($S(a, b)$ denotes the substitution of a by b for any a and b).
- *Partial substitution*: a crowd activity ca_i is used to form a selective control relationship with a service s_i (or a subprocess) under condition C, i.e., $(Q^S((s_i, ca_i, C)) > Q^S(s_i)) \rightarrow S(s_i, (s_i, ca_i, C))$.
- *Pre processing*: a crowd activity ca_i is used to process the input of a service s_i or (or a subprocess), i.e., $(Q^S((ca_i; s_i)) > Q^S(s_i)) \rightarrow S(s_i, (ca_i; s_i))$.
- *Partial pre processing*: a crowd activity ca_i is used to process the input of a service s_i or (or a subprocess) under condition C, i.e., $(Q^S((ca_i; s_i, C)) > Q^S(s_i)) \rightarrow S((s_i, (ca_i; s_i, C)))$.
- *Post processing*: a crowd activity ca_i is used to process the output of a service s_i (or a subprocess), i.e., $(Q^S((s_i; ca_i)) > Q^S(s_i)) \rightarrow S(s_i, (s_i; ca_i))$.
- *Partial post processing*: a crowd activity ca_i is used to process the output of a service s_i (or a subprocess) under condition C, i.e., $(Q^S((s_i; ca_i, C)) > Q^S(s_i)) \rightarrow S(s_i, (s_i; ca_i, C))$.

We use the example of composite translation service in Section 2 (Figure 1) to explain the above types of forms. For the machine translation service process p, the functional QoS dimensions are fluency and adequacy that consist $Q^S(p)$. If the user's requirement of $Q^S(p)$ is Q_u and $Q_u > Q^S(p)$, then the service process itself cannot meet the user's requirement. In that case, we can introduce crowd activities to improve $Q^S(p)$ to meet the condition $Q^S(p) \geq Q_u$ using one or more approaches from the following possible alternatives.

- Substitute the machine translation service process p with a crowd activity for translation ca to make $Q^S(ca) \geq Q_u$;
- Introduce a crowd activity of modification ca for pre processing the input translation source sentence (e.g., change long sentences into short forms or change the sequences of words to be handled by translation service more easily) into machine translation service process p to make $Q^S((ca; p)) \geq Q_u$;
- Introduce a crowd activity of modification ca for post processing the output translation result (e.g., improve the fluency of the result by a monolingual user) with condition $C = (Q^S(p) \geq Q^*)$ into machine translation service process p to make $Q^S((p; ca, C)) \geq Q_u$.

Figure 2 shows an example of service process for translation that combines a composite Web service for translation and crowd services. *Composite Translation Service* indicates the atomic machine translation service or composite machine translation service provided on the Language Grid as shown in Figure 1. For crowd activities, monolingual users and bilingual users can be considered in the translation processes: monolingual users modify the translation results of the machine translation services, while bilingual users confirm the modification results and also translate the contents that cannot be modified by the monolingual users. Two types of HITs (human intelligence tasks)[3] are distributed for crowdsourcing.

4 Experiment

To observe and analyze how crowds could improve service processes, we conduct a large scale experiment for language translation. Our experiments are based on the translation processes that are composed by crowd services and Web services described in Section 3. Settings of QoS measurement, processes, process instances, Web services and crowd services are as follows.

QoS in the language service domain consists of non-functional QoS dimensions (cost, execution time, etc.) and functional QoS dimensions (translation quality: fluency and adequacy). In this experiment, we want to study how crowd services can help improve the non-functional QoS while keeping the same functional QoS. Therefore, we mainly evaluate the execution cost (in $US\$$) and execution time (in *minute*) of the processes.

[3] HIT is used by Amazon Mechanical Turk for description of a human task in the crowdsourcing. We borrow this concept when describing a crowdsourcing task.

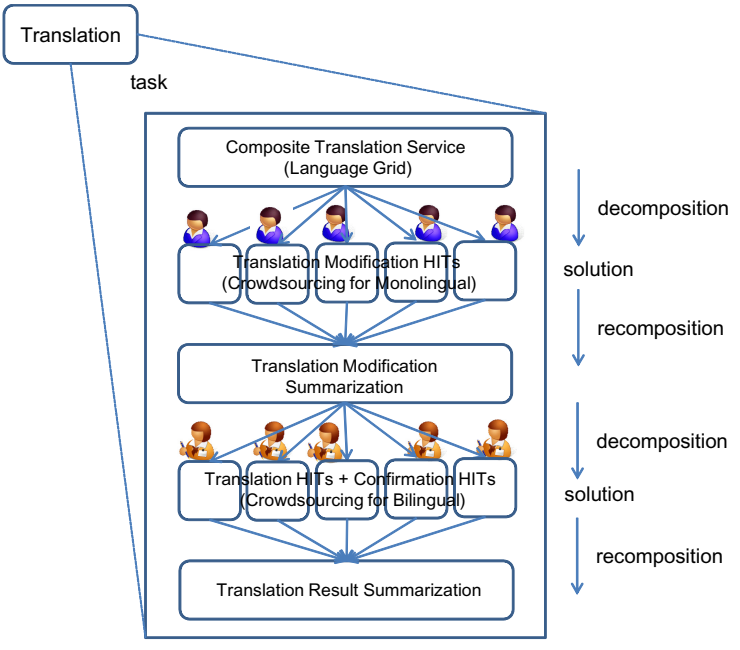


Fig. 2. An example of service process for translation that combines a composite Web service for translation and crowd services.

Main Web services are provided in the Language Grid, including machine translation services, morphological analysis services and dictionary services.

- Machine translation services: Web services by wrapping language resources of JServer machine translation service (Japanese (ja) ↔ English (en), Japanese (ja) ↔ Korean (ko), Japanese (ja) ↔ Simplified Chinese (zh-CN) and Japanese (ja) ↔ Traditional Chinese (zh-TW)) provided by Kodensha Co., Ltd, GoogleTranslate translation service (English (en) ↔ Traditional Chinese (zh-TW)) provided by Google, WebTranser machine translation service (English (en) ↔ German (de), English (en) ↔ French (fr), English (en) ↔ Spanish (es), and English (en) ↔ Portuguese (pt)) provided by Cross Language Inc.
- Morphological analysis services: Web services by wrapping language resources of Mecab Japanese morphological analysis service provided by NTT Communication Science Laboratories, TreeTagger English morphological analysis service provided by University of Stuttgart.
- Dictionary services: dictionary service for Business, University and Temple provided by Kyoto Information Card System Limited Liability Company, Ritsumeikan University and Kodaiji Temples.

Two types of crowdsourcing tasks are included in the experiment, conducted by translation modification monolingual users and translation/confirmation bilingual

users. To study how the quality of crowd activities affects QoS of service processes, we use two different settings for the two types of crowdsourcing tasks.

- Crowdsourcing for monolingual users: low requirements for doing the translation modification tasks. We accept any of the several hundred of registered foreign student users within Kyoto University, Japan. The only requirement is that the registered user is a native speaker of the required modification language. Therefore, the quality of monolingual crowd activity is unanticipatable before the experiment.
- Crowdsourcing for bilingual users: extremely high requirements for doing the translation/confirmation tasks. Only registered users who are of the translation expert level for the required two languages are accepted to do the tasks. Therefore, the quality of the bilingual crowd activity is guaranteed before the experiment.

Table 1. The 14 translation processes used in the experiments that combine Web services (MT: machine translation service; Dic: bilingual dictionary service; MA: morphological analysis service) and crowd activities (Mono: monolingual human service; Bi: bilingual human service).

Process	HITs	Services in the Process				
		MT	Dic	MA	Mono	Bi
Process (1)	551	JServer	Business	Mecab	en	ja,en
Process (2)	551	JServer	Business	Mecab	zh-CN	ja,zh-CN
Process (3)	551	JServer	Business	Mecab	ko	ja,ko
Process (4)	551	WebTranser	Business	TreeTagger	de	en,de
Process (5)	551	GoogleTranslate	Business	TreeTagger	zh-TW	en,zh-TW
Process (6)	551	WebTranser	Business	TreeTagger	pt	en,pt
Process (7)	1,084	JServer	Univeristy	Mecab	en	ja,en
Process (8)	1,084	JServer	University	Mecab	zh-CN	ja,zh-CN
Process (9)	201	JServer	University	Mecab	ko	ja,ko
Process (10)	179	JServer	Temple	Mecab	en	ja,en
Process (11)	179	JServer	Temple	Mecab	zh-CN	ja,zh-CN
Process (12)	179	JServer	Temple	Mecab	ko	ja,ko
Process (13)	179	WebTranser	Temple	TreeTagger	de	en,de
Process (14)	179	WebTranser	Temple	TreeTagger	fr	en,fr

Table 1 shows the 14 service processes of translation. The control flow of each process has been shown in Figure 2. Composite translation service in these processes has been developed with WS-BPEL specification [7] on the Language Grid[4]. Each process is realized by describing the human tasks in BPEL4People [8] and revising the composite translation service in the Language Grid. For example, **Process (1)** in Table 1 is a process for translating

[4] http://langrid.org/service_manager/language-services/profile/TranslationCombinedWithBilingualDictionary

business related documents from Japanese to English. There are altogether 551 process instances in the experiment, each of which represents the task of translating a Japanese sentence to an English sentence. That means 551 HITs are available for crowdsourcing. The composite translation service uses three atomic services on the Language Grid: the JServer Japanese-English machine translation service, the business bilingual dictionary service, and the Mecab Japanese morphological analysis service. Crowdsourcing tasks include translation modification HITs for English monolingual users and translation/confirmation HITs for Japanese/English bilingual users.

5 Observation and Analysis

Based on the experimental settings, we conduct several measurements to analyze how crowd activities affect QoS of the service processes. We first define three indexes that are directly related to the quality of the crowdsourcing tasks: submission rate, acceptance rate and completion rate.

- Monolingual Submission Rate (MSR): the percentage of submitted HITs in all the HITs for monolingual users.
- Monolingual Acceptance Rate (MAR): the percentage of accepted HITs in all the submitted HITs for monolingual users.
- Monolingual Completion Rate (MCR): the percentage of completed HITs (submitted and accepted) in all the HITs for monolingual users, which can be computed by $MCR = MSR \times MAR$.

The reason why we define the three index only for monolingual users lies in that the quality of bilingual users are totally guaranteed before the experiments as described in Section 4.1. Therefore, the submission rate, acceptance rate and completion rate can be regarded as 100% for bilingual users in this experiment.

5.1 Effects of Crowd Activities on Execution Time

We mainly measure the following items to study the effects of crowd activities on execution time of the service process.

- Monolingual Work Time (MWT): execution duration of the crowdsourcing tasks for monolingual users.
- Bilingual Work Time (BWT): execution duration of the crowdsourcing tasks for bilingual users.
- Total Work Time (TWT): summation of monolingual work time (MWT) and bilingual work time (BWT), which can be represented as $TWT = MWT + BWT$.
- Common Work Time (CWT): execution duration when the process is a common human translation process.
- Time Reduction Rate (TRR): the percentage of execution time reduction when comparing with the common human translation process, which is calculated by $TRR = 1 - \frac{TWT}{CWT}$.

Table 2. Measurements of execution time for the 14 service processes

Process	MSR	MAR	MCR	MWT	BWT	TWT	CWT
Process (1)	47.77%	92.13%	44.01%	21.43 min	102.86 min	124.29 min	166.29 min
Process (2)	46.04%	48.39%	22.28%	23.60 min	128.57 min	152.17 min	116.57 min
Process (3)	94.01%	95.30%	89.59%	25.71 min	16.29 min	42.00 min	116.57 min
Process (4)	100.00%	63.53%	63.53%	27.60 min	53.14 min	80.74 min	116.57 min
Process (5)	78.89%	25.32%	19.97%	35.14 min	137.14 min	172.28 min	116.57 min
Process (6)	99.63%	54.07%	53.87%	34.86 min	53.14 min	88.00 min	116.57 min
Process (7)	52.79%	61.40%	32.41%	18.91 min	129.09 min	148.00 min	127.27 min
Process (8)	71.79%	24.52%	17.60%	17.03 min	101.82 min	118.85 min	78.18 min
Process (9)	98.70%	38.73%	38.23%	22.50 min	56.25 min	78.75 min	78.15 min
Process (10)	45.59%	27.33%	12.46%	19.44 min	213.33 min	232.77 min	166.67 min
Process (11)	40.08%	83.43%	33.44%	14.44 min	120.00 min	134.44 min	133.33 min
Process (12)	75.28%	87.60%	65.95%	22.78 min	60.00 min	82.74 min	133.33 min
Process (13)	95.01%	64.32%	61.11%	19.44 min	60.00 min	79.44 min	133.33 min
Process (14)	90.82%	78.45%	71.25%	26.67 min	60.00 min	86.67 min	133.33 min

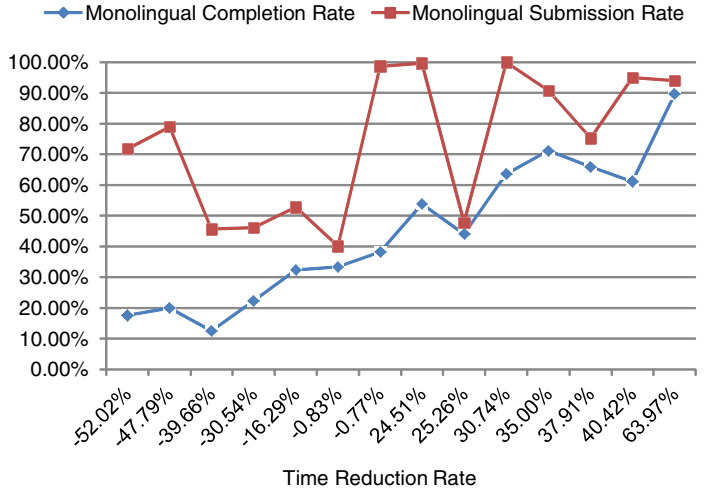


Fig. 3. Relationship between time reduction rate, monolingual submission rate and monolingual completion rate

Table 2 shows the result related to execution time of all the processes in the experiment. All the data is based on the average calculation of one A4 size paper (about 700 Japanese characters or 400 English words) translation. Moreover, monolingual submission rate, monolingual acceptance rate and monolingual completion rate are also acquired, which are based on the average data of all monolingual users who are involved in each process. From the result in Table 2, we can see that the qualities of crowdsourcing tasks of the processes differ much with each other. With the participation of the crowdsourcing activities,

the execution time of the translation task is reduced in half of the 14 processes and is increased in another half.

Figure 3 gives a deeper analysis of the relationship between time reduction rate, monolingual submission rate and monolingual completion rate. The result shows that high monolingual submission rate does not necessarily lead to high time reduction rate. However, there is a trend that higher monolingual completion rate leads to more reduction of execution time comparing with the common human translation process. Table 2 and Figure 3 also shows that it might be difficult to reduce execution time when monolingual submission rate is relatively high while monolingual completion rate is low (**Process (5)**, **Process (8)** and **Process (9)**). The reason lies in that there is much waste of time to deal with the submissions with low quality that are not accepted.

5.2 Effects of Crowd Activities on Cost

To study the effects of crowd activities on execution cost of the service process, we conduct the following measurements. In this experiment, bilingual users and monolingual users are paid by US$50.00 and US$5.00 per A4 page respectively. However, the payment is cut down to half for the HITs that are not accepted.

- Monolingual Work Cost (MWC): cost of the crowdsourcing tasks for monolingual users, which is calculated by $MWC = 5.00 \times (MCR + \frac{1}{2}(MSR - MCR))$.
- Bilingual Work Cost (BWC): cost of the crowdsourcing tasks for bilingual users, which is calculated by $BWC = 50.00 \times (1 - MCR)$.
- Total Work Cost (TWC): summation of monolingual work cost and bilingual work cost, which can be represented as $TWC = MWC + BWC$.
- Common Work Cost (CWC): work cost when the process is a common human translation process, and $CWC = 50.00$.
- Cost Reduction Rate (CRR): the percentage of cost reduction when comparing with the common human translation process, which is calculated by $CRR = 1 - \frac{TWC}{CWC}$.

Figure 4 shows the relationship between execution cost (monolingual work cost (MWC), bilingual work cost (BWC), total work cost (TWC)) and monolingual completion rate. The result shows that composite process by crowdsourcing activities and Web services is possible to reduce the translation cost comparing with the common human translation cost, which supports the analysis in our previous preliminary experiments [5]. The reason lies in that parts of the work in the translation process is substituted with Web services and monolingual users with lower cost. Moreover, the result also shows that the cost reduction rate (CRR) becomes higher if the monolingual completion rate is higher. An extreme successful example is **Process (3)**. Its cost reduction rate reaches 80.41% because of the high quality of the monolingual crowd activity with the monolingual completion rate 89.59%. The trend of the relationship will not change even if we change the unit cost of monolingual users and bilingual users, or change the calculation method of cost.

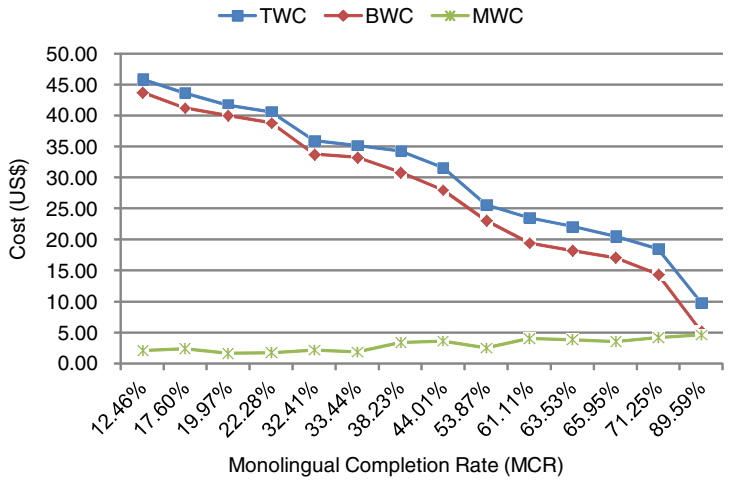


Fig. 4. Relationship between execution cost (monolingual work cost (MWC), bilingual work cost (BWC), total work cost (TWC)) and monolingual completion rate

5.3 Discussion

In this paper, our case study and experiments are in the language service domain. However, the result has its generality since it can reveal the relationship between crowd activities and QoS of service processes. The spectrum of process ranges from service process to human process. The non-functional QoS of service process is generally high (with low execution cost and execution duration) because of its automation and efficiency. However, functional QoS is rather limited because of the application-specific factors. On the other hand, human process can obtain higher functional QoS but lower non-functional QoS, because human can handle the application-specific limitations flexibly with the loss of efficiency. Using monolingual users and bilingual users to compensate the limitations of machine translation services is a good example. To cover both dimensions of QoS, composition of crowd activities and Web services can be regarded as a promising approach.

Based on the analysis, we find out that composition of crowd activities and Web services is possible to be utilized in two directions: composing crowdsourcing tasks into service-based processes and composing services into human-based crowdsourcing activities. For service processes with limited functional QoS like machine translation service, crowd activities can be introduced to improve the functional QoS according to users' requirements. For human-based crowdsourcing activities, it is also feasible to introduce Web services even with limited functional QoS to increase the efficiency to improve non-functional QoS while keeping high functional QoS. However, both directions are currently lack of theoretical and practical foundations because evaluation and prediction of functional QoS and non-functional QoS will be more difficult if there are human activities. Therefore, it is important to consider how to aggregate different dimensions of

QoS from both human activities and Web services when utilizing the composition. Our work can be regarded as a preminalary study in the directions.

6 Related Work

Web service composition has been an important issue for past several years in the service-oriented computing area. Recently, QoS-aware service composition has become the focus in this area [1,2,3]. Zeng *et al.* [1] propose a multidimensional QoS model for Web service composition including dimensions of execution price, execution duration, reputation, successful execution rate and availability. We also use QoS dimensions like execution cost and execution duration as the QoS dimensions in our work. In workflow management area, human task has been studied. From the perspective of link of organization elements and business process, Zhao *et al.* [9] propose a formal model of human workflow based on BPEL4People specifications. They use CSP process algebra to model a human workflow consisting of basic elements of business process engine, task engine and people. However, they do not cover composition of human and Web services. There are also other researches on human workflow from the perspective of organization management [10] and resource management [11]. Comparing with their work, we introduce human activities into service processes from the perspective of QoS. Our research uses the crowdsourcing approach for improving QoS of service processes and conduct large scale experiments in the real world for analyzing the composition of crowd activities and Web services.

7 Conclusion

This paper proposes an approach of composing crowd activities and Web services considering both functional QoS and non-functional QoS of service processes. We conduct a large scale experiment in the domain of language translation to show that composing crowd activities and Web services brings variety and creativity to the traditional service processes and human processes. Further, we analyze the effects of crowd activities on QoS of service processes. We find out that crowd activities with high quality can significantly improve various QoS dimensions of the service processes, while crowd activities with low quality might have negative effects to the service processes. Our future work will mainly focus on unifying crowd services and Web services for solving issues in service composition. We also aim at designing the coordination mechanisms for managing crowd activities in service processes, and controlling the interaction between crowds and services.

Acknowledgments. This work was supported by Strategic Information and Communications R&D Promotion Programme (SCOPE) of the Ministry of Internal Affairs and Communications of Japan.

References

1. Zeng, L., Benatallah, B., Ngu, A., Dumas, M., Kalagnanam, J., Chang, H., Center, I., Yorktown Heights, N.: QoS-aware middleware for web services composition. IEEE Transactions on Software Engineering 30(5), 311–327 (2004)
2. Aggarwal, R., Verma, K., Miller, J., Milnor, W.: Constraint driven web service composition in METEOR-S. In: Proceedings of 2004 IEEE International Conference on Services Computing (SCC 2004), pp. 23–30 (2004)
3. Cardoso, J., Sheth, A., Miller, J., Arnold, J., Kochut, K.: Quality of service for workflows and web service processes. Web Semantics: Science, Services and Agents on the World Wide Web 1(3), 281–308 (2004)
4. White, J., O'Connell, T., O'Mara, F.: The ARPA MT evaluation methodologies: evolution, lessons, and future approaches. In: Proceedings of the First Conference of the Association for Machine Translation in the Americas, pp. 193–205 (1994)
5. Lin, D., Murakami, Y., Ishida, T., Murakami, Y., Tanaka, M.: Composing human and machine translation services: Language grid for improving localization processes. In: Calzolari, N., et al. (eds.) Proceedings of the Seventh International Conference on Language Resources and Evaluation (2010)
6. Ishida, T.: Language grid: An infrastructure for intercultural collaboration. In: IEEE/IPSJ Symposium on Applications and the Internet (SAINT 2006), pp. 96–100 (2006)
7. Alves, A., Arkin, A., Askary, S., Barreto, C., Bloch, B., Curbera, F., Ford, M., Goland, Y., Guızar, A., Kartha, N., et al.: Web services business process execution language version 2.0. OASIS Standard 11 (2007)
8. Kloppmann, M., Koenig, D., Leymann, F., Pfau, G., Rickayzen, A., von Riegen, C., Schmidt, P., Trickovic, I.: Ws-bpel extension for people–bpel4people. Joint white paper, IBM and SAP (2005)
9. Zhao, X., Qiu, Z., Cai, C., Yang, H.: A Formal Model of Human Workflow. In: IEEE International Conference on Web Services, ICWS 2008, pp. 195–202 (2008)
10. Zur Muehlen, M.: Organizational management in workflow applications–issues and perspectives. Information Technology and Management 5(3), 271–291 (2004)
11. Russell, N., van der Aalst, W.M.P.: Work Distribution and Resource Management in BPEL4People: Capabilities and Opportunities. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 94–108. Springer, Heidelberg (2008)

Performance Analysis and Problem Determination in SOA Environments

Vijay Mann[1], Venkateswara R. Madduri[1], and Srividya Shamaiah[2]

[1] IBM Research - India
{vijamann,vmadduri}@in.ibm.com
[2] IBM India Software Labs, Java Technology Center (JTC)
sshamaia@in.ibm.com

Abstract. SOA environments are typically characterized by large number of frameworks. These frameworks stack over each other in the runtime infrastructure and result in deep stack depths and large number of objects being created, most of which, are short lived. Consequently, problem determination and performance tuning of such runtime environments is known to be an extremely difficult task, which requires experience and expertise. In this paper, we share our experiences working with such production SOA runtime environments. Through our experiences we try to find the answer to the following question: can problem determination and performance analysis itself be offered as a service in SOA environments? We note that, in practice Java language and the associated J2EE stack remains one of the most popular runtime environment for implementing SOA. Since Java provides structured runtime logs, we seek to find patterns in those logs that can be used to automate performance analysis and problem determination in SOA environments. We describe three performance problem case studies, each of which present unique performance problems in open source benchmark and production SOA applications. All the case studies highlight the complexity associated with automated performance analysis. However, we make the case that at least part of the performance analysis process can be automated and offered as a service.

Keywords: problem determination, root cause analysis, performance tuning, garbage collection tuning, service performance tuning.

1 Introduction

Service oriented architecture (SOA) runtime environments are often characterized by large number of frameworks. These frameworks stack over each other in the runtime infrastructure. This results in deep stack depths (it is common to find stack depths of 100 or more in such environments) and large number of objects being created, most of which, are short lived. This, in turn, leads to peculiar performance problems in SOA runtime environments which are hard to diagnose. Consequently, problem determination and performance tuning of such runtime environments is known to be an extremely difficult task, which requires experience and expertise.

G. Pallis et al. (Eds.): ICSOC 2011, LNCS 7221, pp. 307–320, 2012.

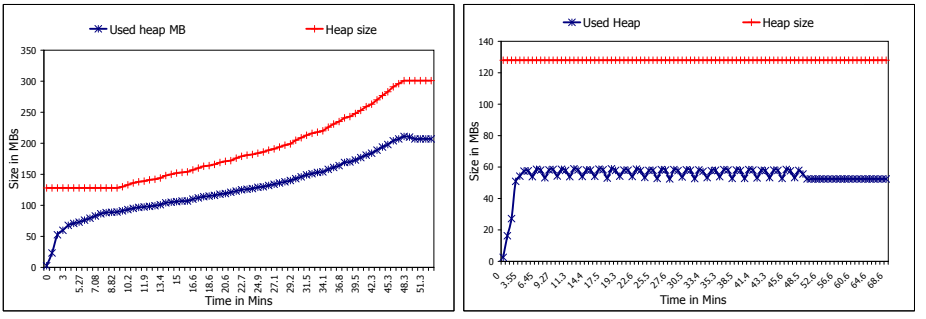
In this paper, we share our experiences working with such production and benchmark SOA runtime environments. Through our experiences we try to find the answer to the following question: can problem determination and performance analysis itself be offered as a service in SOA environments? We note that, in practice Java language and the associated J2EE stack remains one of the most popular runtime environment for implementing SOA. Since Java provides structured runtime logs, such as verbose GC logs and heap dumps, we seek to find patterns in those logs that can be used to automate performance analysis and problem determination in SOA environments. In our interactions with application development, testing and deployment teams, we found that while there was a wide array of performance tuning tools available, the knowledge and skill to use them well was not as common. Furthermore, a large time was being spent in the initial analysis of the problem, most of which can be automated. We describe three performance problem case studies, each of which present unique performance problems in an open source benchmark and a production telecom SOA application. These case studies cover a wide spectrum of performance issues: unavailability due to a memory leak in infrastructure code, high system CPU due to bad developer code, and high CPU and memory usage caused by an application design issue that got aggravated due to incorrect runtime policies and tuning. All the case studies highlight the complexity associated with automated performance analysis. However, we make the case that at least part of the performance analysis process can be automated and offered as a service.

The rest of this paper is organized as follows. Section 2 presents our first case study from an open source benchmark. Sections 3 and 4 describe the second and third case studies from a production telecom enterprise application. An overview of related research is given in Section 5. We summarize our findings and conclude in Section 6.

2 Case Study 1: Memory Leak in Infrastructure Code

In this section we present our first case study - an open source benchmark called "RUBiS" [7]. RUBiS is an open source benchmark that mimics an auction site. It comes with its own workload driver, and has a web tier that connects to a DB tier. It has been cited heavily in research papers for performance evaluation [9]. While using RUBiS as a benchmark for our other research, we noticed that RUBiS would stop responding at the end of an experiment and had to be restarted for the next experiment. We were using the Servlets version of RUBiS and MySQL JDBC driver. We analyzed the verbose GC log of the application for a constant load of 100 clients. It revealed that the used heap kept growing all the time even at constant load (refer Figure 1(a)). This seemed like a classical memory leak issue. However, as we found out, the root cause of this memory leak was not as obvious as most classical memory leaks.

We took two heap dumps of this application within minutes of each other at constant load and compared them. A quick look at the count of objects in those two heap dumps pointed us to a set of objects that have grown almost twice



(a) With memory leak (b) After fixing memory leak

Fig. 1. RUBiS heap usage at constant load

Table 1. RUBiS objects with most growth

Count	Total-size	Type	Growth/Shrink %(by count)
13,743	879,552	com/mysql/jdbc/PreparedStatement$ParseInfo	93.53
13,744	4,288,128	com/mysql/jdbc/PreparedStatement	93.52
13,727	3,294,480	com/mysql/jdbc/ResultSetImpl	93.21
25,644	4,308,192	com/mysql/jdbc/Field	90.43
42,398	2,374,288	java/util/TreeMap	89.98
74,948	2,398,336	com/mysql/jdbc/ByteArrayRow	89.81
394,707	146,608,536	byte[ ]	88.6
84,891	4,074,768	java/util/TreeMap$Entry	80.9
35,851	1,062,672	int [ ]	70.22
162,422	53,270,328	char[ ]	55.64
166,765	6,670,600	java/lang/String	54.48
60,047	2,401,880	java/util/ArrayList	32.87
63,871	2,005,904	java/lang/Object [ ]	30.35
69,266	4,433,024	java/util/HashMap	2.58
69,731	2,868,872	java/util/HashMap$Entry []	2.46
12,519	1,001,520	com/mysql/jdbc/ConnectionPropertiesImpl$Boolean ConnectionProperty	0
31,460	1,258,400	java/util/Hashtable$HashtableCacheHashEntry	-4.92
428,033	17,121,320	java/util/HashMap$Entry	-11.45
22,517	1,441,088	org/apache/tomcat/util/buf/ByteChunk	-16.93
19,856	1,111,936	org/apache/tomcat/util/buf/CharChunk	-17.02
17,646	1,552,848	org/apache/tomcat/util/buf/MessageBytes	-17.11

their original count and size. Most of these objects were of primitive types, but a few (and the ones that had grown the most - more than 90%) were related to PreparedStatement calls in JDBC. A quick breakdown of the objects that grew the most is given in Table 1.

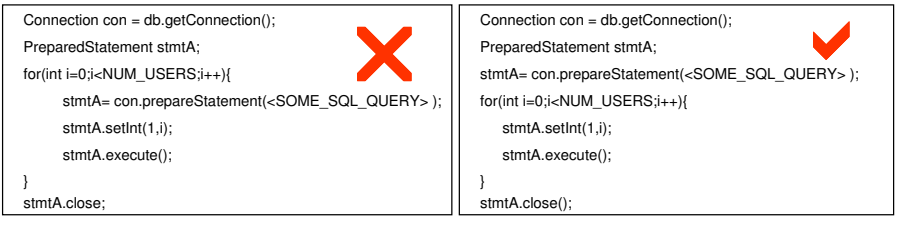


Fig. 2. RUBiS code leaking memory

Since we had the source code of the application with us, we searched for the usage of PreparedStatement objects and we found the pattern shown in Figure 2. Recall that a PreparedStatement object represents a precompiled SQL statement with IN parameters that can be set each time for with the specified setter methods. This way, it can be reused to execute the SQL statement multiple times efficiently. However, RUBiS code repeatedly created a lot of "temporary" PreparedStatement objects inside a loop, but closed only the instance that was created the last. JDBC specification [4] states that closing a JDBC driver should close all PreparedStatements associated with a Connection when the Connection is closed. In this case, the JDBC driver did not close the PreparedStatement objects at Connection close and maintained a reference resulting in those objects not getting garbage collected and their number kept increasing, which resulted in a memory leak.

There are two ways to fix this kind of a leak - either the statement close method is moved inside the loop or the statement creation is moved above the loop. Both will fix the memory leak, but the latter is likely to give better performance. We found that 7 out of 22 servlets in RUBiS were leaking memory at various places due to this usage pattern. The resulting heap usage graph at constant load after fixing the memory leak is given in Figure 1(b). Note that the used heap remains almost constant as expected.

This case study highlighted a memory leak scenario in infrastructure code where the leak happened due to a combination of bad usage as well as bad driver implementation. Furthermore, the root cause turned out to be something that is not usually perceived as a common cause of a leak [2]. This case study provides the following performance problem pattern:

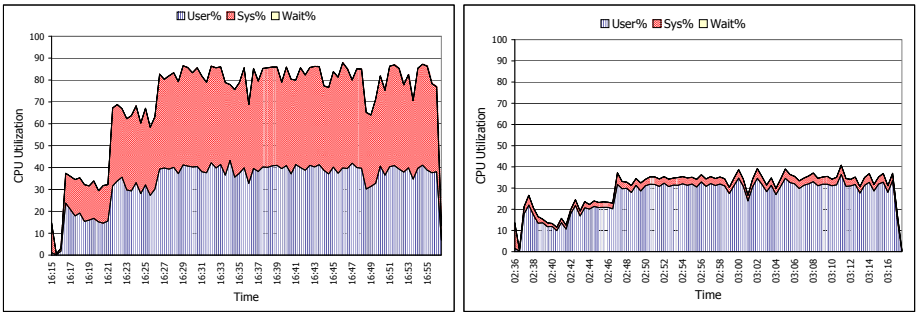
Performance Pattern 1: JDBC PreparedStatements can lead to memory leaks if each instance is not individually closed

This case study also highlighted how automated analysis of verbose GC log at constant load can detect a memory leak. A subsequent automated analysis of multiple heap dumps can be used to pinpoint the objects that are the source of this leak. The final step in the root cause detection - pin pointing the exact code block that is behind the leak, will probably still require an expert to look at the application code or thread dumps.

3 Case Study 2: High System CPU Usage

This section presents our second case study, from a SOA telecom application in production (referred to as TelecomApp in the rest of this paper). This application served as the home page for millions of mobile phone users. Users could click and get various multimedia content such as ring tones, wallpapers, live scores, etc. In production, the main components of this application consist of a cluster of web portal nodes that connect to a cluster of backend content management system nodes, which in turn get their data from a database running on a large multicore machine.

The problem that was reported was of very high CPU utilization at low to moderate number of clients. A quick look at the CPU logs revealed that the web portal node had high levels of system CPU (refer Figure 3(a).)



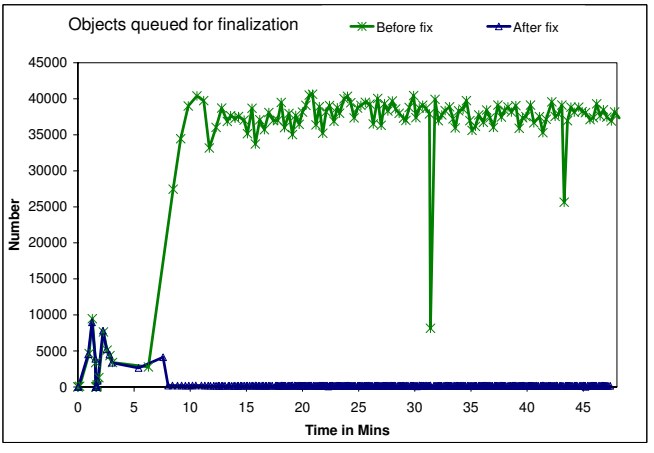
(a) Before Fix (b) After Fix

Fig. 3. Telecom App high system CPU problem

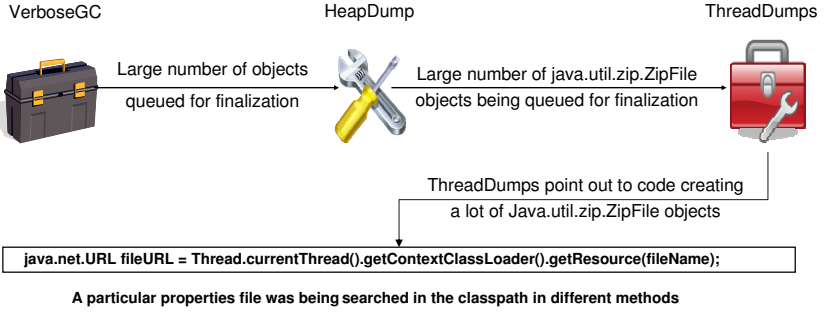
We analyzed the verbose GC logs available. GC logs looked fine except a very high number of objects being queued for finalization - close to 40000 of them (refer Figure 4(a)). Recall that all those objects that have a finalize() method and are found to be unreachable (dead) by garbage collector, are pushed into a finalization queue. At some point later, the finalizer thread will dequeue this object from the finalization queue and call its finalize() method. Too many objects being queued for finalization can cause two types of performance degradation: high CPU utilization because of the finalizer thread's work and/or high memory utilization as all the objects that are dead but reachable from an object that is yet to be finalized, can not be reclaimed.

We took a heap dump in the staging environment to find out the type of objects that were being finalized. Most of the objects that were in the finalization queue were of type "java.util.zip.ZipFile". We took threadumps that pointed us to code that was responsible for creation of these objects of type "java.util.zip.ZipFile". The code invoked the "getResource(Filename)" method on the class loader to get handle to the exact path of a particular properties file. This method, essentially, unzipped all jar files in the classpath, one by one, to search for the given filename. This resulted in creation of a lot of java.util.zip.ZipFile objects (each of which have a finalize method) and a lot of disk read activity. This unzipping of jar files and the associated disk read activity was the reason behind the observed large system CPU. This code was repeated in several methods, all of which, were on the critical path of various client requests. Steps taken to detect the root cause for this problem are shown in Figure 4(b).

We fixed this problem by replacing the code that searched for this jar file in the classpath, by the exact file location of the given file. Immediately, the system CPU went down (refer Figure 3(b)). The number of objects queued for finalization went down as well (refer Figure 4(a)). Note the initial spike in the



(a) Finalization activity in TelecomApp GC logs



(b) Steps taken to resolve TelecomApp high system CPU problem

Fig. 4. Root cause analysis of TelecomApp high system CPU problem

objects queued for finalization in both the cases - this happens as a result of all the classes being loaded from various jars at startup.

This case study highlighted a high system CPU usage scenario where the problem happened due to bad code. This case study provides the following performance problem pattern:

Performance Pattern 2: High Java object finalization activity could be a result of related high filesystem/disk activity.

This case study also highlighted how automated analysis of verbose GC log could have pointed us to abnormal finalization activity. A subsequent automated analysis of a heap dump could have been used to pinpoint the type of objects that were being finalized. The final step in the root cause detection - pin pointing the exact code block that caused the abnormal finalization activity or the high system CPU usage, would probably still have required an expert to look at the application code or thread dumps.

4 Case Study 3: High CPU and Memory Usage

In this section we present our third case study from the same TelecomApp that was described in case study 2. In this case study, we tried to solve the problem of high memory usage by the application coupled with high CPU utilization. This resulted in the application running out of heap memory after running for two or three days and halting with a heap dump getting generated. Application used the default Optthroughput GC policy in the IBM JDK which is aimed at optimal throughput.

We started with the production verbose GC log and looked for the used heap. As shown in Figure 5, the used heap curve (blue line) shows the used heap never goes below 750 M of data (even at night). The used heap becomes the lowest at night (between 1:30-6:00 am).

Logs also had a very high number of mark stack overflow errors - this happens when there are too many live objects in the heap (or more precisely very deeply nested objects) and the stack that GC uses during the mark phase overflows. Figure 5 also shows two restarts indicated by the sudden drop in heap size.

Verbose GC log also showed high pause times due to high compaction times. This pattern was very uniform though out and almost all high pause times were caused due to high compaction times (refer Figure 6). Further analysis of the logs for these high pause times, revealed allocation failures with reason code 16. Reason code 16 compaction happens when the garbage collector is not able to increase the amount of free storage by at least 10% [3]. The overhead due to garbage collection was around 13% (GC overhead=times spent in GC pauses/-total execution time * 100). Figure 6 also shows that GC keeps freeing memory - however the freed memory (green curve) keeps decreasing due to fragmentation and eventually compaction is required (blue lines). Note that heap fragmentation can also lead to "dark matter" [5] which can not be used for satisfying allocation requests. Any heap space that lies between two allocated objects and is less than 512 bytes, is not used by the JVM for allocation requests and is termed as "dark matter".

An analysis of the production heap (that got dumped on an out of memory error) showed a total of 1.2 GB of memory being used out of which close to 585 MB of the retained heap is occupied by objects belonging to two classes (refer Table 2):

1. com.TelecomApp.cache.impl.CacheEntryImpl, and
2. com.TelecomApp.mcs.devices.InternalDeviceImpl.

Note that retained size of an object is its shallow size plus the shallow sizes of the objects that are accessible, directly or indirectly, only from this object [6]. Table 3 shows the break down of the shallow heap by object types. Recollect that shallow size of an object is the amount of memory allocated to store the object itself, not taking into account the referenced objects. One can see that the top 3 contributors are all related to HashMaps. "java.util.HashMap$Entry" and "java.util.HashMap$Entry[]" contribute nearly 390 MBs (close to 33% of total heap size). This hints that both com.TelecomApp.cache.impl.CacheEntryImpl

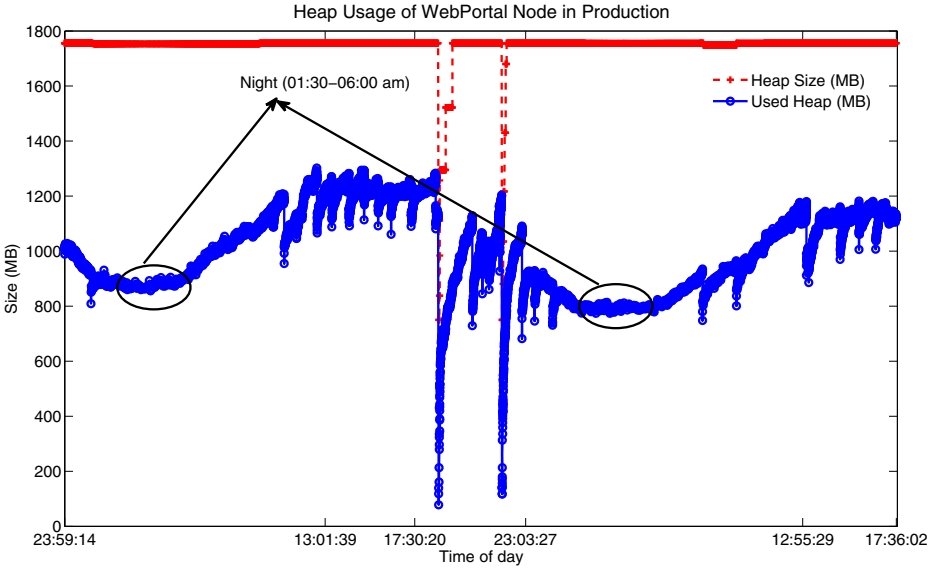


Fig. 5. Continuous high memory usage in the TelecomApp

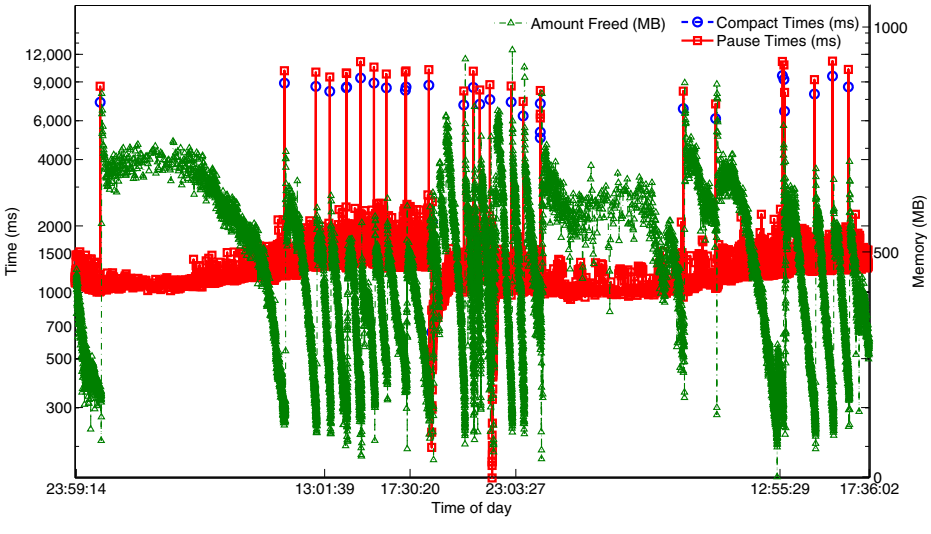


Fig. 6. TelecomApp: Pause times, compaction times and memory freed

and com.TelecomApp.mcs.devices.InternalDeviceImpl comprise of large Hashmaps. The application development team reconfirmed that the application internally uses a lot of caches to cache mobile device specific policies for faster response time. This explained, why the heap usage never goes down drastically. This seemed like an application design issue since the memory footprint of the

Table 2. TelecomApp production heap dump breakup (sorted by retained heap size)

Type	Count	Shallow Heap (MB)	Retained Heap (MB)	Retained Heap (%)
com.TelecomApp.mcs.devices.InternalDeviceImpl	8,907	0.680	418.6	33.96
com.TelecomApp.cache.impl.CacheEntryImpl	197	0.011	165.9	13.46
com.ibm.ws.cache.Cache	70	0.022	63.8	5.18
com.ibm.ws.util.ThreadPool$Worker	224	0.029	60.6	4.92
com.ibm.ws.webcontainer.webapp.WebApp	160	0.032	42.2	3.42
com.ibm.ws.webcontainer.httpsession.MemorySessionData	1,402	0.182	40.9	3.32
java.lang.Class	33,406	3.380	29.5	2.39
oracle.jdbc.driver.T4CPreparedStatement	425	0.425	27.4	2.22
com.ibm.ws.webcontainer.httpsession.MemorySessionContext	160	0.035	19.4	1.57
java.lang.String	160,233	4.890	18.3	1.48
com.TelecomApp.mcs.accessors.jdbc.JDBCDeviceRepositoryAccessor	1	0.000	15.3	1.24
com.ibm.wps.state.outputmediators.OutputMediatorFactoryProxy	96	0.003	13.6	1.1
Remainder	25,159,774	1,223	317	25.72

Table 3. TelecomApp production heap dump breakup (sorted by shallow heap size)

Count	Total-size	Type
1,871,075	305,453,504	array of char
8,558,144	273,860,608	java/util/HashMap$Entry
927,421	114,439,696	array of java/util/HashMap$Entry
1,774,002	56,768,064	java/lang/String
639,453	52,688,552	array of java/lang/Object
92,922	49,771,728	array of byte
901,808	43,286,784	java/util/HashMap
899,467	35,978,680	com/TelecomApp/mcs/themes/PropertyValue
109,740	27,300,328	array of int
26,622	16,834,352	array of com/TelecomApp/mcs/themes/StyleValue
514,376	16,460,032	java/util/Hashtable$Entry
329,675	15,824,400	com/TelecomApp/styling/properties/PropertyDetailsImpl
398,769	12,760,608	com/TelecomApp/styling/impl/engine/StylerImpl
251,689	11,886,160	array of com/TelecomApp/mcs/themes/PropertyValue
324,782	10,393,024	com/TelecomApp/mcs/css/version/DefaultCSSProperty
340,836	9,177,496	array of long
95,246	8,393,000	array of java/util/Hashtable$Entry
338,905	8,133,720	java/util/BitSet
491,191	7,859,056	java/util/HashSet
281,817	7,144,296	array of com/TelecomApp/styling/impl/engine/matchers/SimpleMatcher
284,648	6,831,552	com/ibm/ws/cache/Bucket
203,781	6,520,992	javax/servlet/jsp/tagext/TagAttributeInfo
175	5,708,528	array of org/apache/xpath/objects/Xobject
234,155	5,619,720	com/TelecomApp/styling/impl/device/DeviceStylesDelta
221,120	5,306,880	java/util/ArrayList

application seemed to be more than what the system could provide. However, the application team maintained that cached entries should automatically expire every 30 minutes or so and they ruled out a redesign since it would have required considerable effort.

Verbose GC logs also pointed that the application was using some large objects. We analyzed the heap dump for the shallow object sizes. There were only 30,000 objects greater than 1024 bytes in a total of close to 25 million objects. This constitutes almost 0.12% of the heap in terms of number of objects. However, the cumulative space occupied by these large objects was roughly 222 MB out of the total 1.2 GB (18.5%). The average object size was 51 bytes, while the median size was 32 bytes which was also the mode. This indicated that heap comprised of a very large number of small objects which were interspersed throughout the heap in between few large objects. A histogram of the shallow objects is given in Figure 7.

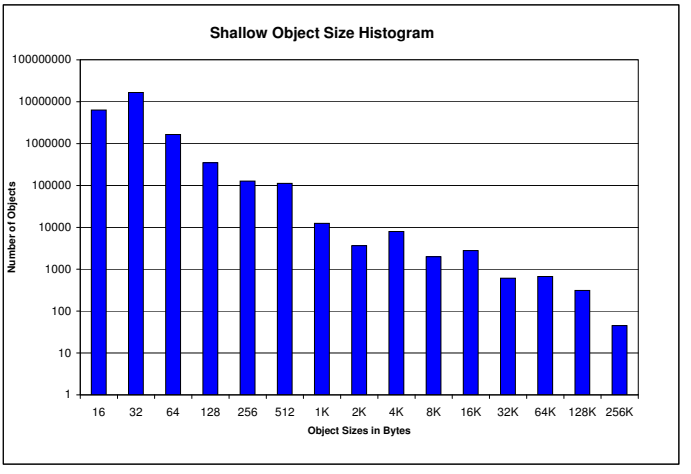


Fig. 7. Histogram of object sizes from the production heap

In order to rule out any memory leaks, a long 10 hour run at low and constant load was conducted in the staging environment. The heap usage from the verbose GC log was almost constant through out the experiment which ruled out any memory leaks. However, the staging environment could not be used to replicate the long pause times observed in the production system. This was probably due to the load offered to the system.

Our analysis so far ruled out any memory leaks and the following characteristics became apparent:

- the heap was fragmented and it resulted in high compaction times and long periods of sustained high CPU usage, as well as wasted heap space
- the application had a mix of a few large and a lot of small objects
- the application had some short-lived objects and most of the heap seemed to comprise of relatively long living cached objects.

Garbage collection techniques to resolve the above issues have been discussed in literature. Given the above application characteristics, we considered two solutions:

1. **Using a generational and concurrent collector**: This policy (known as the gencon GC policy in IBM JDK) is usually recommended for transactional applications that create a lot of short-lived objects. This policy divides the heap space into nursery (for new and short lived objects) and tenured space for old objects. Since this policy uses copying of live objects in the nursery space during a collection in the nursery (minor collection), it gets rid of fragmentation in the nursery. However,when the tenured space fills up, a global collection occurs and unreachable objects are collected, which may result in fragmentation in the tenured space. Ideally we would have liked to move all the long lived cached objects into the tenured space and leave the nursery only for short lived objects.

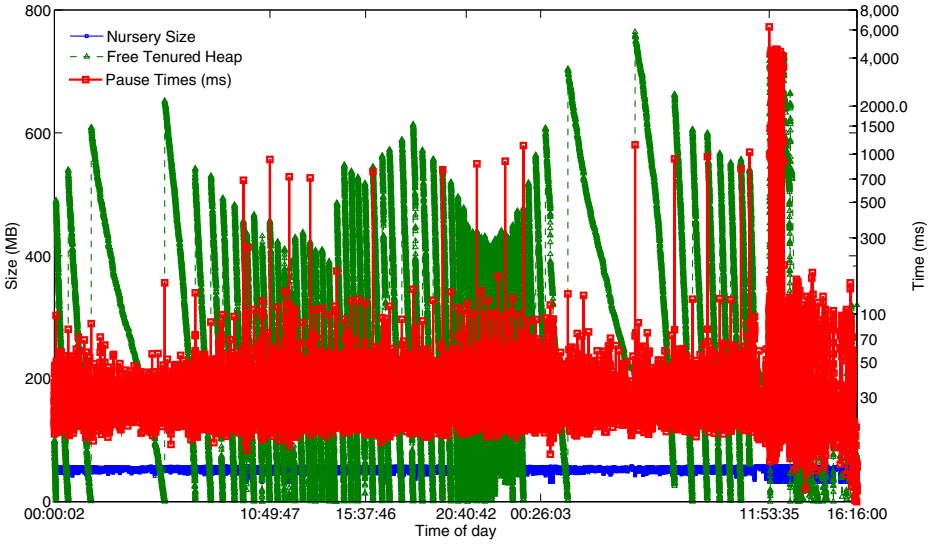


Fig. 8. TelecomApp with modified GC policy (Gencon) reduced pause times but small nursery size resulted in frequent global collections (saw-tooth pattern)

2. **Using a pool of objects** for the most frequently created objects. This resolves the problem of fragmentation by reducing creation and collection of short lived objects. However, this requires either creating a pool for frequently created objects in the application code or using the inbuilt pooling capability in the IBM JVM [1].

We tried both solutions, but realized that the first option was easier to implement on a production system. We first tried the generational and concurrent collector with default settings. Garbage collection overhead (percentage of time spent in GC pauses) went down from 13% to 1.18%. In the default gencon setting, the size of the nursery heap is set to 55-60 MB and the rest is used as the tenured space. When this setting was put into production, the high pause times went away as there was no compaction required (refer Figure 8). However, a saw-tooth pattern in the used memory and free tenured space (refer Figure 8) also indicated that the default nursery size was too small. This resulted in too many minor collections, and objects being promoted into tenured space prematurely. This also resulted in the tenured space filling up very soon and requiring a global collection.

The saw-tooth pattern revealed that around 400-550 MB of memory was being freed from the tenured space in each global collection. We used this as a yardstick to set the initial size of nursery heap to be 256 MB and the maximum nursery size to be 512 MB. When this setting was implemented in production the GC overhead further went down from 1.18% in the default gencon settings to 0.8% (refer Table 4). The problem of high CPU usage for a sustained period

Table 4. TelecomApp: performance improvement through GC tuning

GC policy	Time spent in GC pauses	Max pause time in seconds	Mean pause time in seconds
Optthroughput GC policy (original)	13.07%	11.2	1.35
Gencon GC policy (default nursery size)	1.18%	6.26	0.03
Gencon GC policy (increased nursery size)	0.8%	1.18	0.08

of time (which was being caused by frequent fragmentation and compaction in the default settings) was also resolved.

This case study highlighted a high CPU usage scenario where the problem happened due to a combination of application design and infrastructure settings (garbage collection policy settings). The application suffered high GC overheads due to heap fragmentation and repeated compaction and this resulted in high CPU utilization. This case study provides the following performance problem pattern:

Performance Pattern 3: Sustained levels of high CPU even in the absence of high load could be a result of heap fragmentation

This case study also illustrated the complexity associated with performance tuning: optimally tuning GC settings for an enterprise application that had an interesting mix of objects (mix of large and small objects, long living and short living objects) can be non-trivial and hard to automate. However, a few steps could have been automated: An automated analysis of verbose GC logs could have pointed us to high pause times which were being caused by high compaction times. Automated analysis of the heap dumps would have pointed to the existence of a large number of short-lived objects and a rule based analysis could have recommended the right GC policy for this scenario.

5 Related Research

Performance analysis and problem determination of enterprise applications is a broad area. Some performance problems are caused due to resource bottlenecks such as insufficient CPU or memory, or incorrect runtime infrastructure settings such as incorrect thread pool or connection pool sizes. These problems are somewhat independent of the nature of application and there has been a lot of work in automating problem determination and performance analysis of such problems. This typically involves statistical analysis of performance or monitoring data from middleware systems [12,8] to detect any changes or anomalies in server performance.

On the other hand, analysis of performance problems that are closely tied to the application design, the application code, application memory usage characteristics and the interaction of the application with its runtime are much harder

to resolve and automate. The case studies presented in this paper belong to this category of performance problems. In a recent work, Chis et. al [10] present a solution that discovers a small set of high-impact memory problems, by detecting patterns within a Java heap. They demonstrate that eleven patterns cover most memory problems, and that users need inspect only a small number of pattern occurrences to reap large benefits. While this work mainly focuses on memory footprint issues and overhead of meta data in various Java collection data structures, it shares our view of using a repository of patterns to automate performance analysis.

There has also been a lot of recent work done on automating detection of memory leaks in Java programs [13, 14, 11]. While, we made use of standard heap comparison techniques to detect a memory leak in our first case study, these newer techniques can further make the process simpler and quicker and these can be part of an integrated automated peformance analysis service.

6 Conclusion

In this paper we presented our experiences working with enterprise SOA Java applications that exhibited various performance and scaling problems. We shared the insights we got while determining the root cause of some of these performance problems. We diagnosed the availability problem with RUBiS, an open source benchmark, and found the root cause to be a memory leak that occurred due to usage error as well as a non-compliant JDBC driver implementation. We presented two other case studies from a production telecom SOA application. One of them resulted in high CPU utilization caused due to code that searched a properties file in the entire classpath. This manifested itself as high number of finalization objects that get created when various jar files are unzipped. The last case study demonstrated various garbage collection issues that modern SOA applications in Java face. Through our analysis we were able to recommend the correct GC policy and tune its settings. Throughout our journey of fixing these performance problems, analysis of verbose GC logs and heap dumps proved to be valuable diagnosis tools that gave us vital clues on the probable root causes. These logs were readily available from production systems.

Our experience, also demonstrated, that the underlying causes of these problems tend to be diverse and it is very hard to provide an end-to-end root cause analysis engine for these problems. In our interactions with application development teams, we observed that they do not seem to have all the insights about the performance implications of their code and the testing and deployment teams did not have the required skill and experience to diagnose the problems from GC and heap logs. We believe that a performance analysis service for SOA environments that automates the analysis of verbose GC logs and heap dumps, and determines a high level root cause based on performance patterns such as those provided by us, can be of great value.

Acknowledgements. We would like to thank Ravi Kothari and Manish Gupta from IBM Research - India, Ashish Agrawal from IBM Global Business Services, Rajeev Palanki from IBM Java Technology Center, and Suparana Bhattacharya from IBM India Software Labs who provided us with valuable suggestions, guidance and much needed support throughout this work.

References

1. Heap fragmentation with IBM 1.3.1 and 1.4.2 JVMs, http://www-01.ibm.com/support/docview.wss?uid=swg21196072
2. Java Diagnostics Guide 1.4.2 - Common causes of perceived leaks, http://publib.boulder.ibm.com/infocenter/javasdk/v1r4m2/topic/com.ibm.java.doc.diagnostics.142/html/commoncausesofleaks.html
3. Java Diagnostics Guide 1.4.2 - verbose GC outut from a compaction, http://publib.boulder.ibm.com/infocenter/javasdk/v1r4m2/index.jsp?topic=/com.ibm.java.doc.diagnostics.142/html/id1156.html
4. JDBC 4.0 API Specification Final Release - JSR-000221, http://java.sun.com/products/jdbc/download.html
5. Mash that trash – Incremental compaction in the IBM JDK Garbage Collector, http://www.ibm.com/developerworks/ibm/library/i-incrcomp/index.html
6. Retained and Shallow Heap, http://www.yourkit.com/docs/90/help/sizes.jsp
7. RUBiS Homepage, http://rubis.ow2.org/
8. Agarwal, M.K., Sachindran, N., Gupta, M., Mann, V.: Fast Extraction of Adaptive Change Point Based Patterns for Problem Resolution in Enterprise Systems. In: State, R., van der Meer, S., O'Sullivan, D., Pfeifer, T. (eds.) DSOM 2006. LNCS, vol. 4269, pp. 161–172. Springer, Heidelberg (2006)
9. Cecchet, E., Marguerite, J., Zwaenepoel, W.: Performance and Scalability of EJB Applications. In: ACM OOPSLA (November 2002)
10. Chis, A.E., Mitchell, N., Schonberg, E., Sevitsky, G., O'Sullivan, P., Parsons, T., Murphy, J.: Patterns of Memory Inefficiency. In: Mezini, M. (ed.) ECOOP 2011. LNCS, vol. 6813, pp. 383–407. Springer, Heidelberg (2011)
11. Jump, M., McKinley, K.: Cork: dynamic memory leak detection for garbage-collected languages. In: Symposium on Principles of Programming Languages, POPL (2007)
12. Mann, V., Agarwal, M.K., Gupta, M., Sachindran, N.: Problem Determination in Enterprise Middleware Systems Using Change Point Correlation of Time Series Data. In: IEEE/IFIP NOMS (2006)
13. Mitchell, N., Sevitsky, G.: Leakbot: An Automated and Lightweight Tool for Diagnosing Memory Leaks in Large Java Applications. In: Cardelli, L. (ed.) ECOOP 2003. LNCS, vol. 2743, pp. 351–377. Springer, Heidelberg (2003)
14. Xu, G., Rountev, A.: Precise memory leak detection for java software using container profiling. In: ACM International Conference on Software Engineering, ICSE (2008)

Approaches to Improve Reliability of Service Composition

Jörg Hohwiller, Diethelm Schlegel, and Gregor Engels

Capgemini, CSD Research, Berliner Str. 76, 63065 Offenbach, Germany
{joerg.hohwiller,diethelm.schlegel}@capgemini.com, engels@upb.de

Abstract. Nowadays, enterprises realize functionality as systems that are composed of services. This includes even mission critical parts of their business. Hence, the reliability of such systems including their composition and services is increasingly important. However, it is a challenge to establish a high reliability in this context because distribution of functionality increases the potential points of failure. Different approaches exist to increase reliability but they typically act on a restricted scope like network layer or software design. In order to obtain better results, it is often necessary to combine multiple approaches depending on the actual situation and the requirements. This paper classifies commonly used approaches according their scope and rates their effects on the reliability. Thereby, it supports the selection of approaches to improve reliability and finally helps to find a suitable solution for a given situation.

Keywords: Reliability, SOA, Composition, Service, Quality of Service.

1 Problem

Services are functionalities that are exposed over networks using standardized interfaces. They are the base of *service oriented architectures* (SOA). Enterprises implement even critical business logic by services. In the past, the majority of services was implemented internally. Nowadays, it can be observed that the use of external services is growing fast. On the other hand, *service composition* combines multiple existing services to a new one which offers a more complex functionality. As shown in figure 1, the realization uses a wide range of different hardware and software components including networks. [3] deals with the prediction of the resulting reliability of the composite service.

Independent of whether a service is composed or not, consumer expect its reliability in accordance with the concrete usage conditions. Generally, *reliability* is defined as the ability of a service to perform its required functions under stated conditions for a specified time interval [10] (e.g. an availability of 99.45% for a 7/24 service typically means it must be fully functional at least 363 days a year). It contains four aspects: maturity, fault tolerance, recoverability, and reliability compliance (see [7]).

Service level agreements (SLA) typically regard reliability with respect to *stated conditions*. To guarantee a SLA, systems must cope with this well defined

G. Pallis et al. (Eds.): ICSOC 2011, LNCS 7221, pp. 321–332, 2012.

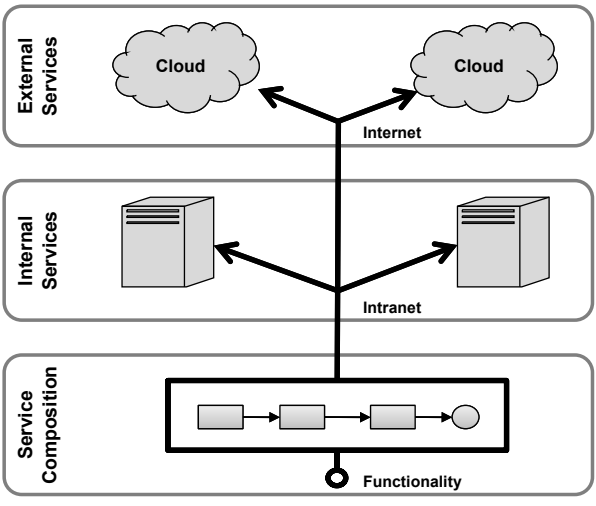


Fig. 1. Service Composition

set of *situations* that could be systematic or accidental, temporary or permanent, targeted or unintended. The importance of conditions depends on the concrete usage scenarios. Several users of the same service may even have different SLAs depending on their actual conditions and payments.

In many cases, the effects of service faults on the business are disastrous. So, measures must be taken for reducing the risk of service failures. There are several commonly used techniques like failover mechanisms or redundancy to enhance service reliability. But their value depends on the environment and situation. When designing a system, architects must be familiar with these interrelations to choose the right concepts. For example, it has no major effect to replicate databases when the most critical problems arise from overloaded networks.

In the following sections, situations with negative impact on the behaviour of composed services are classified. Some approaches for reliability improvement are described and rated concerning their effects in these situations. This will give architects an advice to decide which techniques he shall use for individual usage scenarios and possible fault situations. Hereby, our results mainly result on empirical experiences collected in customer projects.

2 Situations and Scopes

The overall reliability of a composed service is affected by a large variety of situations. This section introduces two dimensions that classify situations into scopes. Thus, it is possible to rate the capabilities of approaches to increase the service reliability in individual situations (see section 3).

The first dimension consists of different levels. of influence It ranges from the central service composition up to external services in the cloud. In the same way, the approaches to improve the reliability act on one or more of these levels. The

following levels are distinguished in this paper. They correspond with the three layers represented in figure 1.

- The *composition level* summarizes the logical design, the model, and the physical implementation of resulting composed services, e.g. by using *business process engines* (BPE). It also includes the interfaces that make the resulting services available.
- The *internal level* contains hardware, network components, operating systems (OS), runtime environments, and the implementation of the services used by the composition.
- The *external level* collects hardware and software components external to the company. The main difference to the first two levels consists in the fact that there are only few or no possibilities to influence these components.

As a second dimension, the situations are grouped by provoking causes leading to deteriorated reliability of service compositions. There are four condition clusters regarded in this paper.

- *Overloads* are characterized by the presence of too many or too complex requests. This implies that components are used more intensely than they are designed for. If the situation is provoked intentionally it is called a *denial of service attack*.
- *Failures* sum up all situations where hardware or software components do not perform the way they are designed to. Examples are unstable operating systems, software abortions, or complete breakdown of hardware (servers, network, power-supply, internet connection, etc.).
- As opposed to failures, *design mistakes* cover all situations where components sure perform well but their behaviour is not consistent with their requirements. They are more systematic and occur deterministically. Reliable service compositions are expected to handle such conditions properly.
- *Attacks* contain situations where components are out of order due to external or internal enemies aimed at bringing wohle systems or single components down.

Table 1 shows examples of situations in order to clarify the two dimensions described above. The same table layout will be used in section 3 to rate the approaches.

3 Solution

This section shows approaches aimed at improving the reliability of services and compositions. The solution is created by choosing the right combination of approaches for the given requirements. Each approach is summarized by a table showing the effects on reliability for the classes as in table 1 (see section 2). Each cell shows a rating of the impact for the corresponding situations. A gain is marked as +. On the other hand a - indicates a negative and an empty cell no relevant influence.

Table 1. Situations with Impact on Reliability

Level \ Cond.	Overload	Failure	Design	Attack
Composition	▪ Too many requests ▪ Too complex models	▪ Model abortion ▪ Race condition ▪ Dead lock	▪ Wrong system architecture ▪ Incorrect service composition	▪ Distributed Denial of Service ▪ Unauthorized access
Internal	▪ Too many invocations ▪ CPU load too high	▪ Server crash ▪ Power outage ▪ Ressource leak ▪ Service failure	▪ Wrong config ▪ Insufficient hardware ▪ Unsuitable network architecture	▪ Trojan Horse ▪ Viruses
External	▪ Not enough bandwidth ▪ Overload of external service	▪ Internet (access) problems ▪ External service not available	▪ Provider SLA not sufficient ▪ Ext. Service not suitable	▪ Distributed Denial of Service ▪ Worms

3.1 Load Balancing

An obvious approach to increase reliability is to have multiple redundant instances. This applies to all technological levels in order to avoid single points of failure.

- *Composition:* multiple instances (physical hardware or virtual instances)
- *Internal services:* multiple instances of internal services, network redundancy
- *External services:* second ISP, redundant cloud services

Having redundant instances available in cold-standby will only help to avoid maintenance downtimes. In undesired conditions (failure and overload) availability can be increased by dynamic delegation of requests across multiple available redundant instances. This is achieved by a proxy as front-end to the service. Such proxy serve two demands.

- *Failover:* If one of the redundant instances becomes unavailable requests are delegated to another instance.
- *Load balancing:* The proxy distributes the load of incoming requests among the available instances. There are various strategies like round-robin [2].

There are existing hardware and software solutions to realize *load balancers* (proxy servers capable of load balancing and failover). Especially, this applies for HTTP as a de facto standard for the transport of services. The same approach can be used on external level in the other direction by clustering internet connections of different service providers.

A redundant design of service implementations can cause problems if the functionality is not read only. Write operations allowing modifications of underlying persistent data require synchronization of changes between redundant instances.

This is solved by concepts like master/slave replication or clustering of databases (e.g. see [1]). On the other hand the service logic should be designed stateless for good scalability.

The rating of this approach is summarized in table 2. It improves the reliability only if applied on the composition and on internal services because it is impossible to influence the redundancy the whole external equipment involved. Furthermore, design mistakes and attacks cannot be relieved by duplicating instances.

Table 2. Approach: Load Balancing

Summary:	Dynamically route requests to redundant instances according to load and presence			
Preconditions:	Stateless service or synchronization required			
Advantages:	- Simple and generic approach - Established, evident gain of availability			
Drawbacks:	- High cost (e.g. duplicate hardware costs, increasing maintenance) - Potential synchronization issues on *service level*			
Level \ Cond.	**Overload**	**Failure**	**Design**	**Attack**
Composition	+	+		
Internal	+	+		
External				

3.2 Quality of Service

As described in section 1 the requirements for reliability differ per usage scenario. Here we extend the approach *redundancy* (see 3.1). Instead of giving all requests the same priority in the load balancing, the quality of service (QoS) level depends per usage scenario. This mechanism can be applied on composition side when exposing the service interface and for the internal and external services. QoS can be achieved by different approaches.

– *QoS per Endpoint (Virtualized Services)* defines different endpoints (URLs) for each usage scenario that represent the same service logic but offer different QoS. Realizing this with entirely redundant hardware would be a waste of resources. Instead, the endpoints represent *virtualized services* that delegate to the same back-end (the redundant instances described in section 3.1). Each virtualized service is realized as separate load balancer (see 3.1) and can ensure a different QoS level. This also includes limiting the QoS by reducing the number of parallel requests or not involving all available redundant instances from the backend (see Figure 2). Different usage scenarios then use different virtual services. On *infrastructure level* it can be ensured that only the dedicated virtual service is reachable for a particular client. In advance to load balancing and failover (see 3.1) this approach is more robust. Overload conditions and attacks will only block a single usage scenario.

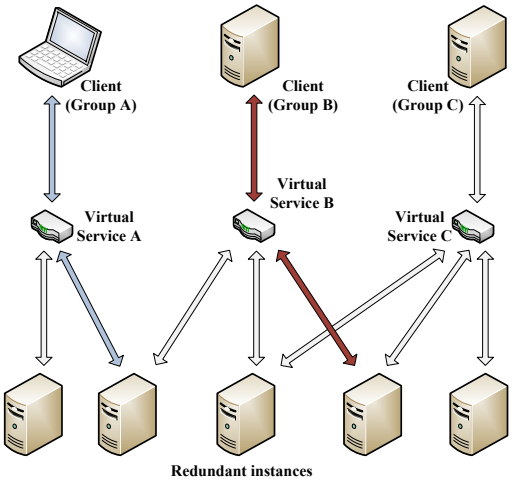


Fig. 2. QoS per Endpoint

- Typically, the user is related to the usage scenario that is associated to a particular QoS level (*QoS per User*). The user invoking the service needs to be identified and its according QoS level determined. If the service requires authentication the user identification arises implicit. Otherwise information as the IP-address have to be used. As such information can be manipulated additional security mechanisms may be required. If the appropriate QoS level is determined, the application logic (service facade) can route the request accordingly to one of the redundant back-end instances (e.g. analogue to *QoS per endpoint*). This should typically be addressed by an enterprise service bus (ESB). However, it is important to assure the chosen product is robust enough not to become a bottleneck.
- A very flexible approach is to send the QoS requirements with each service request (*QoS per Request*, see also [13]). The question is how much trust shall be given to the requester. In a closed and isolated world this approach will work well, but in an open and uncontrolled world (e.g. the public cloud) some control and restriction mechanisms have to be added on the provider side. Otherwise a DoS-attacker will have it even easier to produce high load. The main scenario of this approach is therefore a service that is used by a smaller set of clients over a secured channel. If requests are charged according to the required QoS a balance will establish. Such offering will typically require service level agreements and measurement (see [5] and [11]) for proper acceptance.

Table 3 shows the rating of this approach. Compared with failover mechanisms it copes with denial of service attacks and overload situations because only single QoS groups are affected. This even is relevant for external services. As a drawback, establishing appropriate QoS based redundancy is more expensive than simple failover mechanisms.

Table 3. Approach: Quality of Service

Summary:	Dynamically route requests to redundant instances with different QoS-levels			
Preconditions:	Redundancy (load balancing and failover)			
Advantages:	- Availability even in case of overload / attack			
Drawbacks:	- High cost (e.g. multiple times of hardware costs, more maintenance) - Potential synchronization issues on service level			
Level \ **Cond.**	**Overload**	**Failure**	**Design**	**Attack**
Composition	+	+		+
Internal	+	+		+
External	+			+

3.3 Parallel Service Invocation

If the reliable service needs some functionality provided by another service and there is more than one possible service provider offering the needed function, the requesting service can decide what service it calls.

The service calls at least two of the possible providers, waits until the first result is received and aborts the other calls. If data is modified it is important that changes are performed only once. This implies the need for compensation, transactional behaviour or two phases commit (see [12]). The call is successful if at least one service provider is performing well. The time needed for the call is the minimum time of all providers (see [9] and [8]).

The more independent the service implementations are in respect of infrastructure and implementation, the higher will be the benefit of this approach. A disadvantage is the higher resource utilization. The network load increases by sending multiple requests in parallel. If adopted on the large scale service providers must cope with more requests in a given time interval. The approach is not useful against attacks and concerning the composed service itself.

3.4 Dynamic Service Composition

The main disadvantage of the redundant service invocation approach consists in the load it generates and the need for compensation or two phases commits. It is desirable to call only one of the service providers to save bandwidth. As a solution, *static service composition* is established by manually configuring the provider to use. This approach resembles the load balancing approach (see 3.1) with another service provider available as a backup.

As a drawback, failure and overload cannot be detected without delay and intervention. The solution is to choose the provider dynamically. The challenge is to find the service provider with the best reliability based on an algorithm. It must react on changing conditions fast and in an appropriate way. The general problem to find the best provider at any time cannot be solved, because it is

Table 4. Approach: Parallel Service Invocation

Summary:	Call more than one service provider in parallel			
Preconditions:	Critical service, ≥2 providers available, compensation or two phase commit, stateless service			
Advantages:	- Guarantees the best response behaviour at the given conditions			
Drawbacks:	- Generates overload on network, proxies and service implementation			
Cond. / Level	**Overload**	**Failure**	**Design**	**Attack**
Composition				
Internal	-	+		
External	-	+	+	

impossible to predict the future conditions. Besides, regular checking of service availability would increase network load.

Optimizing dynamic service composition is known as NP-hard (see [14]). However, there are a lot of proposals for QoS aware service compositions based on integer programming, case-based reasoning, genetic algorithms (see [4]), and hybrid approaches (see [14]). Which of these will be appropriate depends on the given situation and structure. Sudden overload, attacks, or failure conditions will be recognized with delay and therefore negative impact on reliability. Additionally, it improves the behaviour in the context of design mistakes (composition and external services).

Table 5. Approach: Service Composition

Summary:	Call the service provider with best predicted QoS			
Preconditions:	Critical service, ≥ 2 providers available, probability for sudden overload or failure not too high			
Advantages:	- Improves the response behaviour by making some assumptions of the service provider conditions			
Drawbacks:	- Reacts on sudden overload or failure with delay			
Cond. / Level	**Overload**	**Failure**	**Design**	**Attack**
Composition			+	
Internal	+	+		+
External	+	+	+	+

3.5 Service Interface Granularity

All preceding approaches take services as given and add the appropriate mechanisms. Another way to improve reliability consists in better system design. It is

the preferred way when implementing a new composition including the internal services, but it can also be accomplished when re-designing consisting applications or selecting external services. This paper considers the service granularity. In any case, extremes must be avoided. Monolithically designed services as well as thousands of small services will lead to poor performance and reliability.

As stated in [6] coarser grained interfaces reduce the number of calls and at the same time network utilization. So, the impact of overload will be minimized. On the other hand, the interfaces will become more complex leading to increasing requirements and execution times as well as to higher design mistake rates. This also decreases the reliability against attacks. The probability to find alternative providers for functionalities decreases.

Finer grained interfaces result in leaner implementations. The chance to find redundant providers becomes higher which indirectly leads to more robust behaviour in overload or failure conditions. Transactional reliability can be achieved more easily if operations are less complex. At the same time, the increasing number of service calls leads to higher network utilization. If external providers are called the individual round trip delays of multiple invocations sum up reducing the performance compared to singe service calls. In summary it can be stated that finer grained services have more demands concerning the underlying network infrastructure.

Table 6. Approach: Service Granularity

Summary:	Choose the appropriate interface complexity and service granularity			
Preconditions:	Service composition not yet fixed			
Advantages:	- More reliable implementation, better chance to find redundant service implementations			
Drawbacks:	- Possible increase of network and proxy utilization			
Cond. / Level	**Overload**	**Failure**	**Design**	**Attack**
Composition	+		+	+
Internal	+		+	
External	+		+	+

3.6 Solutions for Usage Scenarios

Figure 3 shows an overview of the approaches and the resulting positive and negative impact on reliability. This implies solutions for the following usage scenarios of a service

– *Internal:* As long as the system is composed of services in the local network redundancy (3.1, 3.2) will be the adequate approaches to ensure sufficient reliability. Additionally, if the infrastructure is properly sized, it is preferable to choose a finer interface granularity (3.5). If external services are involved

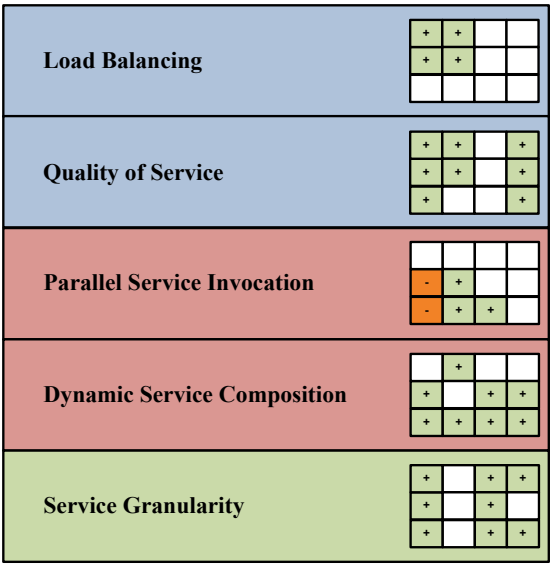


Fig. 3. Approaches to Improve Reliability

the approaches with respect to the parallel service invocation (3.3) or dynamic service composition (3.4) will become preferred solutions.

- *Business to Business:* If multiple partners must be provided by different QoS levels for the same service, one of the QoS based redundancy (3.2, 3.2) is the best approach. The interface granularity should be rather coarser grained (3.5) to reduce the network traffic.
- *Business to Customer:* If the service is also exposed to customers it is highly recommended to choose the QoS per Endpoint approach (3.2). Requests should be separated into QoS groups to be able to guarantee high service reliability for important users. Finer grained interfaces (3.5) would be slightly better in this case to prevent denial of service attacks by calling complex services.

Based on the results there are the following ruler of thumb. The usage scenarios and rules of thumb must be checked against the actual requirements before implementing them because their effects highly depend of the concrete scenario.

1. As long as services are composed internally redundancy should be established.
2. As long as cloud services are involved parallel invocation or dynamic service composition are appropriate.
3. Cloud services should tend to be coarser granular.
4. Internal services should be designed finer granular.

4 Conclusion and Further Work

This paper dealt with the question how to make a service composition more reliable, especially if external service providers are involved. Based on two dimensions several approaches were rated and summarized with respect to their impact on reliability. The concrete solution consists in selecting one or more approaches depending on the actual scenario.

Furthermore, the solution depends on the point of view. A service provider can improve the reliability by load balancing (3.1) and QoS mechanism (3.2) while a service requester will consider parallel invocation or dynamic service composition (3.4). This is summarized by exemplary usage scenarios and rules of thumb in section 3.6. After all, true service reliability can only be achieved if all technological levels are addressed.

Based on this paper possible future work includes analysing further approaches (e.g. architectural design questions) and extending dimensions (e.g. with conditions like duration and severity). Further, the methodology can be extended by incorporating security aspects or implementation costs. So, the ratings of the approaches will become even more useful.

References

1. Database replication, http://www.tech-faq.com/database-replication.html
2. Bhavin Turakhia, P.K.: The compendium of load balancing strategies. Tech. rep., Wiki. Directi (2009)
3. Bocciarelli, P., DAmbrogio, A.: A model-driven method for describing and predicting the reliability of composite services. Software and Systems Modeling 10, 265–280 (2011), doi:10.1007/s10270-010-0150-3
4. Canfora, G., Di Penta, M., Esposito, R., Villani, M.L.: An approach for qos-aware service composition based on genetic algorithms. In: GECCO 2005: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, pp. 1069–1075. ACM, New York (2005)
5. Dobson, G.: Quality of service in service-oriented architectures. Tech. rep., Lancaster University (2004)
6. Friedl, B.: Zur optimalen Granularität von IT-Services - Eine Analyse relevanter ökonomischer Einflussfaktoren. In: Bernstein, A., Schwabe, G. (eds.) Proceedings of the 10th International Conference on Wirtschaftsinformatik, vol. 1, pp. 404–413 (2011)
7. International Organization for Standardization: ISO/IEC 9126. Software engineering – Product quality (2001)
8. Kokash, N., D'Andrea, V.: Evaluating Quality of Web Services: A Risk-Driven Approach. In: Abramowicz, W. (ed.) BIS 2007. LNCS, vol. 4439, pp. 180–194. Springer, Heidelberg (2007)
9. Laranjeiro, N., Vieira, M.: Towards fault tolerance in web services compositions. In: Guelfi, N., Muccini, H., Pelliccione, P., Romanovsky, A. (eds.) EFTS, p. 2. ACM (2007)
10. Lee, K., Jeon, J., Lee, W., Jeong, S.H., Park, S.W.: QoS for web services: Requirements and possible approaches. Tech. rep., W3C, Web Services Architecture Working Group (November 2003), http://www.w3c.or.kr/kr-office/TR/2003/ws-qos/

11. Ludwig, H., Keller, A., Dan, A., King, R.P., Franck, R.: Web service level agreement language specification, 1.0 (2008)
12. May, N.R.: A redundancy protocol for service-oriented architectures (2008)
13. Tian, M., Gramm, A., Ritter, H., Schiller, J.H., Voigt, T.: Qos-aware cross-layer communication for mobile web services with the ws-qos framework. In: Dadam, P., Reichert, M. (eds.) GI Jahrestagung (2). LNI, vol. 51, p. 286. GI (2004)
14. Ye, X., Mounla, R.: A hybrid approach to qos-aware service composition. In: Proceedings of the 2008 IEEE International Conference on Web Services, pp. 62–69. IEEE Computer Society, Washington, DC (2008)

Author Index