Services Science

LNCS 6568

E. Michael Maximilien Gustavo Rossi
Soe-Tsyr Yuan Heiko Ludwig
Marcelo Fantinato (Eds.)

Service-Oriented Computing

ICSOC 2010 International Workshops
PAASC, WESOA, SEE, and SOC-LOG
San Francisco, CA, USA, December 2010
Revised Selected Papers

Springer

Lecture Notes in Computer Science 6568

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

E. Michael Maximilien Gustavo Rossi
Soe-Tsyr Yuan Heiko Ludwig
Marcelo Fantinato (Eds.)

Service-Oriented Computing

ICSOC 2010 International Workshops
PAASC, WESOA, SEE, and SOC-LOG
San Francisco, CA, USA, December 7-10, 2010
Revised Selected Papers

Springer

Volume Editors

E. Michael Maximilien
IBM Almaden Research Center, 650 Harry Road, 95120-6099 San José, CA, USA
E-mail: maxim@us.ibm.com

Gustavo Rossi
University of La Plata, Faculty of Informatics
50 & 115 st., first floor (1900) La Plata, Buenos Aires, Argentina
E-mail: gustavo@lifia.info.unlp.edu.ar

Soe-Tsyr Yuan
National Chengchi University
Taiwan, No. 64, Sec. 2, ZhiNan Rd., Wenshan District
Taipei City 11605, Taiwan, R.O.C.
E-mail: yuans@mis.nccu.edu.tw

Heiko Ludwig
IBM T.J. Watson Research Center
19 Skyline Dr., Hawthorne, NY-10532, USA
E-mail: hludwig@us.ibm.com

Marcelo Fantinato
University of São Paulo
03828-000 São Paulo - SP, Brazil
E-mail: m.fantinato@usp.br

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-19393-4 e-ISBN 978-3-642-19394-1
DOI 10.1007/978-3-642-19394-1
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2011921525

CR Subject Classification (1998): D.2, C.2, H.4, H.3, H.5, J.1

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

Welcome to the workshop program of the 8th International Conference on Service Oriented Computing (ICSOC 2010), held in San Francisco, December 2010. These workshop proceedings represent high-quality research and industry papers that showcase recent and new working developments in service-oriented computing and related fields.

Since the first meeting in Trento in 2003, ICSOC has become the premier conference in the rapidly evolving areas of service research. While keeping its roots in scientific excellence and technical depth of service technology, ICSOC 2010 aimed to combine technical aspects of service computing with application and business-oriented aspects of service design, following the recent emergence of service science as an interdisciplinary foundation for understanding an integrating service systems.

This ICSOC's workshop program comprised four workshops on current topics, presenting results and work in progress: (1) the First International Workshop on Performance Assessment and Auditing in Service Computing (PAASC 2010) addressed among its topics monitoring, compliance verification and the associated costs; (2) the Workshop on Engineering Service-Oriented Applications (WESOA 2010), organized for the sixth time in the context of ICSOC, addressed topics ranging from engineering services for a cloud environment to QoS of services to using "wisdom" as input to service composition; (3) the First International Workshop on Services, Energy, and Ecosystem (SEE 2010) looked at environmental aspects of services, ranging from power consumption of data centers to issues of caloric footprint of services using employees for fulfillment; and, finally, (4) the Second International Workshop on Service-Oriented Computing in Logistics (SOC-LOG 2010) discussed current work on managing logistics services, from infrastructure aspects such as event-driven approaches to issues of multimodal services, addressing the challenges of integrating many participants across organizational and legislative boundaries.

A special thanks goes to the workshop authors, keynote speakers, panelists, and tutorial speakers, who together contributed to this important aspect of the conference. And of course, our biggest thanks goes to you, the tutorial and workshop participants. It is our great pleasure and privilege to present these proceedings. We hope you find it inspiring and useful for years to come.

December 2010

Heiko Ludwig
Fu-ren Lin
Michael Maximilien
Gustavo Rossi
Soe-Tsyr Yuan

Organization

Honorary General Chair

Jim Spohrer	IBM, USA

General Chairs

Heiko Ludwig	IBM Research, USA
Fu-Ren Lin	National Tsing Hua University, Taiwan

Program Committee Chairs

Mathias Weske	University of Potsdam, Germany
Jian Yang	Mcquarie University, Australia
Paul Maglio	IBM Research, USA

Advisory Board

Avi Borthakur	Oracle, USA
Claudio Bartolini	HP Labs, USA
Christoph Bussler	Saba Software, USA
Axel Hochstein	Stanford University, USA
Ramesh Jakka	360 Fresh and Carnegie Mellon Silicon Valley, USA
Yuecel Karabulut	SAP, USA
Michael Maximilien	IBM Research, USA

Workshop Chairs

Gustavo Rossi	UNLP, Argentina
Soe-Tsyr Yuan	National Chengchi University, Taiwan
Michael Maximilien	IBM Research, USA

Tutorial Chairs

Thomas Sandholm	HP Labs, USA
Jyoti Bhat	Infosys, India
Haluk Demirkan	Arizona State University, USA

Demonstration Chairs

Florian Daniel	University of Trento, Italy
Christian Zirpins	Karlsruhe Institute of Technology, Germany

Poster Chair

Andreas Wombacher	University of Twente, The Netherlands

Finance/Registration Chairs

Bernd J. Kraemer	Fernuni Hagen, Germany
Christian Zirpins	Karlsruhe Institute of Technology, Germany

PhD Symposium Chairs

Eleni Stroulia	University of Alberta, Canada
Boualem Benatallah	University of New South Wales, Australia
Jianwen Su	University of California at Santa Barbara, USA

Web Chair

Maja Vukovic	IBM Research, USA

Publications Chair

Marcelo Fantinato	University of São Paulo, Brazil

Publicity Chair

Matthias Weidlich	University of Potsdam, Germany

Conference Operations Management

Beatriz Raggio	Xparency, USA

PAASC Workshop Organizer

Claudia-Melania Chituc	University of Porto, Portugal

WESOA Workshop Organizing Committee

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

SEE Workshop Organizing Committee

Barbara Pernici	Politecnico di Milano, Italy
Schahram Dustdar	Technical University of Vienna, Austria
G.R. Gangadharan	Politecnico di Milano, Italy
Patricia Lago	VU University Amsterdam, The Netherlands
San Murugesan	University of Western Sydney, Australia

SOC-LOG Workshop Organizing Committee

Joerg Leukel	University of Hohenheim, Germany
André Ludwig	University of Leipzig, Germany
Alex Norta	University of Helsinki, Finland

Table of Contents

PAASC 2010 Workshop

WESOA 2010 Workshop

SEE 2010 Workshop

SOC-LOG 2010 Workshop

Introduction to the First International Workshop on Performance Assessment and Auditing in Service Computing (PAASC 2010)

Claudia-Melania Chituc[1,2]

[1] Faculty of Engineering, University of Porto, Department of Informatics Engineering
[2] LIACC – Artificial Intelligence and Computer Science Laboratory, University of Porto
FEUP-DEI, Rua Dr. Roberto Frias, 4200-365 Porto, Portugal
cmchituc@fe.up.pt

Abstract. The main goal of the International Workshop on Performance Assessment and Auditing in Service Computing (PAASC) was to bring together researchers and industry representatives, providing them the opportunity to present research and development results, discus lessons learned (e.g., from designing, building and using services), and advance novel ideas on topics in the area of performance assessment and auditing in service computing. Technical papers of very good quality have been submitted, of which only 5 papers have been accepted.

Keywords: Service computing; performance assessment.

1 Scope

Services are autonomous and platform independent computational entities [1][2]. Service oriented computing brings the promise of assembling application components in a network of services, with applications crossing distributed computing platforms and organizations [3]. In a service computing environment, service providers, service users (clients) and service brokers collaborate and compete, aiming at attaining specific goals.

Service computing provides organizations significant technical support to increase their competitiveness. Although the benefits of service computing are widely recognized, generally accepted analytical models, frameworks, methodologies and metrics to analyze, quantify and monitor the (economic) performance in service computing are not yet available. Research in the area of service computing focuses mainly on technical aspects (e.g., interoperability [4][5]; QoS [6]). Research related to (economic) performance assessment, auditing and monitoring in service computing is scarce. Several research questions in this area continue to be unanswered:

- Which are the business models for service computing?
- Which are the most relevant theories and approaches to support the formal definition and assessment in service computing?

E.M. Maximilien et al. (Eds): ICSOC 2010 Workshops, LNCS 6568, pp. 1–3, 2011.

- How can benefits of service computing be assessed?
- Which are the most relevant metrics to support (economic) performance assessment in service computing?

The scope of this workshop was to address these issues. The main goal of the First International Workshop on Performance Assessment and Auditing in Service Computing (PAASC 2010) was to bring together researchers and industry representatives, providing them the opportunity to present research and development results, discuss lessons learned (e.g., concerning the design, development and use of services), and advance novel ideas on topics in the area of performance assessment and auditing in service computing. Application papers discussing the power and applicability of conceptual methods and frameworks to real-world problems have also been of interest for PAASC 2010 Workshop.

2 Workshop Organizer (and Chair)

Prof. Dr., engineer, economist Claudia-Melania Chituc, Faculty of Engineering, University of Porto, Department of Informatics Engineering (FEUP-DEI) and LIACC.

3 Program Committee

- Armando Colombo, Schneider Electric GmbH, Germany
- Claudio Bartolini, HP Labs, Palo Alto, USA
- Frank Leymann, University of Stuttgart, Germany
- Herve Panetto, Nancy University, France
- Eliot Salant, IBM Haifa Research Lab, Israel
- Hong-Linh Truong, Vienna University of Technology, Austria
- Shimon Y. Nof, Purdue University and PRISM Center, USA
- Florian Rosenberg, CSIRO ICT Centre, Australia
- Francisco Restivo, University of Porto, Faculty of Engineering, Department of
- Informatics Engineering (FEUP-DEI) and LIACC, Portugal
- George Spanoudakis, City University London, UK
- Philipp Leitner, Vienna University of Technology, Austria
- Claudia-Melania Chituc, University of Porto, Faculty of Engineering, Department of Informatics Engineering (FEUP-DEI) and LIACC, Portugal.

References

1. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F.: Service-Oriented Computing: State of the Art and Research Challenges. IEEE Computer, 64–70 (June 2007)
2. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F.: Service-Oriented Computing: A Research Roadmap. International Journal of Cooperative Information Systems 17(2), 223–255 (2008)
3. Leymann, F.: Combining Web Services and the Grid: Towards Adaptive Enterprise Applications. In: Proceedings CAiSE 2005 Workshops, FEUP Edições, vol. 2, pp. 9–21 (2005)

4. Chituc, C.-M.: Interoperability in Collaborative Networks: A Framework Proposal. PhD Thesis in Electrical and Computer Engineering, Faculty of Engineering, University of Porto (2008)
5. Chituc, C.-M., Azevedo, A., Toscano, C.: A Framework proposal for seamless interoperability in a collaborative networked enviornment. Computers in Industry 60, 317–338 (2009)
6. Menascé, D.A.: QoS Issues in Web Services. IEEE Internet Computing, 72–75 (November-December 2002)

A Case Study on Optimizing Web Service Monitoring Configurations

Garth Heward[1], Jun Han[1], Ingo Müller[1], Jean-Guy Schneider[1], and Steve Versteeg[2]

[1] Swinburne University of Technology, Hawthorn, Victoria, Australia
[2] CA Labs, Melbourne, Victoria, Australia
{gheward,jhan,imueller,jschneider}@swin.edu.au,
Steve.Versteeg@ca.com

Abstract. Whilst monitoring web services provides benefits in terms of demonstrating that Service Level Agreements (SLAs) have been met, monitoring comes with a cost on the QoS of delivered services. We present the application of our method for optimizing the configuration of a suite of web service monitors in order to minimise the QoS impacts of monitoring on a web service provider. We discuss the actions required for optimization, and benefits that were achieved. Through this case study, we highlight the possible benefits of optimization to a web service provider, and give details on how we achieve optimization.

Keywords: Web Services, Monitoring, Monitoring Optimization.

1 Introduction

WS (Web Service) providers give quality guarantees for their services, so that consumers are assured of the expected Quality of Service (QoS) of services before and during service consumption. To demonstrate that they have met these guarantees, providers monitor their services. However, achieving this monitoring goal introduces a problem, since monitoring can impact the delivered quality of web services. This impact has been demonstrated to be as much as 40% (for response time) for a single monitor [1]. In order to reduce these impacts, we have developed a method for optimizing a suite of WS monitors [2]. This optimization balances the benefits of monitoring, in terms of monitoring coverage and the costs of monitoring, in terms of QoS impacts.

In this paper, we present an application of our optimization technique on the IT system of a WS provider, in the form of a case study. This case study is designed to demonstrate both the information and procedures required for optimization, and the possible benefits for a WS provider. After presenting an overview of our optimization technique, we describe in detail its application to a realistic WS system. This includes how we determine an optimal configuration for the case study, and the effect that this configuration has on delivered QoS. To achieve this, we measure the effectiveness of the optimization by testing the case study system before and after optimization is applied.

E.M. Maximilien et al. (Eds): ICSOC 2010 Workshops, LNCS 6568, pp. 4–14, 2011.

In Section 2, our case study system to which optimization will be applied is described. In Section 3, our technique for optimization is described. In Section 4, the application of optimization onto the case study is described. In Section 5, a series of experiments applying optimization to the case study system are described and their results are provided and discussed.

2 iTravel Scenario

iTravel is a company that offers web services to travel agencies for booking flights and hotels. iTravel has web services flight_info and hotel_info, used to lookup flight and hotel information, and services book_flight and book_hotel, used to book flights and hotels. iTravel has SLAs with its consumers, giving guarantees for response time, reliability and compliance to privacy regulations. If iTravel is unable to demonstrate that they have met these requirements they must refund affected consumers a percentage of their monthly account fees. iTravel must also meet their corporate governance requirements. These state that iTravel is accountable for the security compliance and privacy compliance for their customers' information. iTravel is liable for large fines if they cannot demonstrate that these requirements have been met.

To demonstrate that the SLA and corporate governance requirements have been met, iTravel has a suite of monitors. iTravel has progressively added monitors into its system, each meeting a new monitoring requirement. iTravel now has a suite of monitors that can measure response time and reliability, and meet security and privacy compliance requirements by detecting security and privacy violations. In recent months, however, iTravel have not met their SLA obligations for response time or reliability, and were required to refund the account fees of affected consumers. Investigation revealed that WS-monitors were imposing a large performance burden on the WS system and the monitors in fact had been doing some overlapping monitoring. Furthermore, the use of some monitors had caused services to become unreliable due to non-response. Therefore, iTravel would like to have their monitors optimally configured so that they only monitor when it is valuable to do so (when requirements demand it), and only the least costly (in terms of QoS) monitor is selected for each monitoring requirement.

2.1 iTravel System

The iTravel system consists of three subsystems, one for each of Flights, Hotels, and Administration. Each subsystem is hosted on its own, dedicated hardware. This case study focuses on the Flights Subsystem, shown in Figure 1. As part of the normal application system, web services are shown in light grey, rounded boxes in the figure; the WS consumers (there may be many) are shown in the light grey, square box; and WS proxies are shown in boxes with dark grey backgrounds. The monitors are shown in numbered, square white boxes, including two Probe type monitors (flight_info Probe and book_flight Probe), two Interceptor type monitors (book_flight Interceptor and hotel_info Interceptor), and one

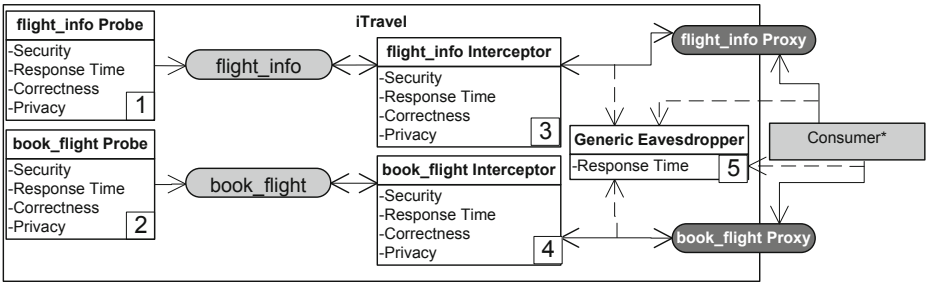


Fig. 1. iTravel Flight Subsystem

Eavesdropper monitor (Generic Eavesdropper). Solid lines represent service invocation/ communications, and dashed lines show points of message interception.

The aspects of the services being monitored concern security, response time, privacy, and reliability. Security monitoring supports the assessment of the compliance with security regulations by continuously producing well-defined and traceable evidence (hereafter simplified to 'Security') by checking messages for known weaknesses such as SQL injection and verifying that actions such as authentication have been performed. Response time monitoring measures the round-trip time for a service request, on the service provider's side. Privacy monitoring supports the assessment of the compliance with privacy compliance regulations (hereafter simplified to 'Privacy') by performing actions such as verifying and cleansing outgoing personal information. Monitoring for reliability checks that requests receive a response, and that the response is properly formatted.

The probe monitors (flight_info Probe and book_flight Probe) are capable of meeting requirements for monitoring response time, reliability, security and privacy of the web services by invoking the services and pretending to be service consumers. The probes are hosted on the same server as the web services.

The interceptor monitors (flight_info Interceptor and book_flight Interceptor) measure the same properties, but do so by intercepting the service requests of consumers, rather than generating their own requests as probes do. The interceptors have the capability to forward messages after analysing them, and therefore interceptors can filter or modify communications and perform functions such as firewalling. The interceptor monitors are hosted on their own, dedicated servers.

The eavesdropper monitor (Generic Eavesdropper) acts similarly to the interceptors; however, it cannot modify or stop a message from being transmitted, as it is only a passive listener. The Generic Eavesdropper is hosted on its own, dedicated server.

3 Overview of Monitoring Optimization

The optimization problem is to find a monitoring configuration that gives the maximum value in terms of QoS impact (costs) and monitoring coverage (benefits). In this section, we present an overview of our optimization approach in

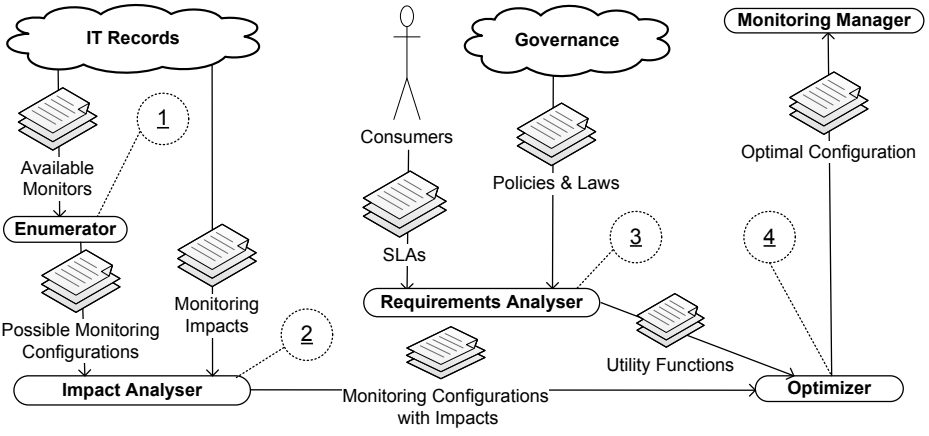


Fig. 2. Optimization Framework

order to describe what actions are performed to calculate an optimal configuration. The full details of this optimization approach can be found in [2,3].

Figure 2 shows a framework illustrating our approach to solving this monitoring optimization problem, annotated with the ordering of activities of the optimization process. In step **1**, the set of deployed monitors is identified based on IT records, and then the Enumerator generates all the possible monitoring configurations for this set of monitors. In step **2**, the performance and quality impacts of each monitor are identified from IT Records or benchmarking, and the Impact Analyser derives the total performance impact of each possible monitoring configuration. In step **3**, the monitoring requirements and associated penalties for not meeting them are identified from analysing SLAs, policies and laws, and the Requirements Analyser transforms requirements into a set of utility functions that define the benefit gained from monitoring a quality whilst a specific level of performance is being achieved. In step **4**, the Optimizer uses the set of utility functions from the Requirements Analyser to score all monitoring configurations with impacts from the Impact Analyser, and the monitoring configuration that yields the highest utility is selected and applied to the monitoring system.

4 iTravel Optimization

In order to obtain the best monitoring coverage whilst delivering acceptable QoS, we have applied optimization as described in Section 3 to the iTravel system as described in Section 2.

4.1 iTravel Baseline QoS

The response time for both the flight_info and book_flight services in the iTravel scenario has been measured under controlled (lab) conditions to be 2.5 seconds. Security and Privacy have been measured to be 100%, and as such are considered

100% as long as there is at least one monitor observing each of them at a sampling rate of 100% - otherwise they are 0%. Analysis of historical executions has been performed to determine that reliability is 98% under ideal conditions of no monitoring for the flight_info service, and 99% for the book_flight service.

4.2 iTravel Requirements

iTravel is bound by SLA and corporate governance requirements. The SLA requirements are from a single SLA that all service consumers have with iTravel, and the corporate governance requirements come from iTravel's business and legal obligations. The requirements on the flight subsystem are:

SLA1. flight_info Response Time will be $\leq$ 10s, penalty=33% account fee
SLA2. book_flight Response Time will be $\leq$ 10s, penalty=66% account fee
SLA3. flight_info will be $\geq$95% Reliable, penalty=100% account fee
SLA4. book_flight will be $\geq$98% Reliable, penalty=100% account fee
SLA5. flight_info will meet Privacy requirements, penalty=100% account fee
SLA6. book_flight will meet Privacy requirements, penalty=100% account fee
CG1. flight_info will meet Security requirements, penalty=$10,000
CG2. book_flight will meet Security requirements, penalty=$10,000
CG3. flight_info will meet Privacy requirements, penalty=$5,000
CG4. book_flight will meet Privacy requirements, penalty=$5,000

For example, SLA1 states that iTravel must deliver a response time of under 10 seconds for the flight_info service, otherwise the consumer must be refunded 33% of their monthly account fee (which is $100).

4.3 iTravel Monitoring Impacts

We divide the types of monitoring impacts into monitoring instance impacts (iMI) and monitoring overhead impacts (iMO). The monitoring instance impacts are those that occur each time a monitor is used. For example, there may be an impact on response time whenever a probe monitor is used to test a service, and the more times the monitor is used, the higher the impact will be. Monitoring overhead impacts differ in that they are a fixed impact that occurs whenever a monitor is enabled. For example, using an interceptor-based monitor may increase response time by a fixed amount (due to extra transmission time), regardless of how many events that monitor observes.

We have measured the monitoring instance impacts and monitoring overhead impacts[1] for response time and reliability in the iTravel system. These impacts are an average percentage increase of response time and decrease of reliability, compared with the response time and reliability of a service without monitoring. For example, using the book_flight Probe will decrease reliability by 4% and increase response time by 2% for the book_flight service, plus an additional 1% on response time for each property measured (security, response time, privacy, or reliability). Additionally, each quality measured with the book_flight Probe impacts

[1] Available at http://www.ict.swin.edu.au/personal/gheward/

the response time of the flight_info service by 0.5%. The interceptor monitors have a fixed impact regardless of how many qualities of service are monitored, e.g. the impact of monitoring security, response time, reliability and privacy of flight_info using the flight_info Interceptor is the same as the impact of monitoring only security. Note that some monitors have an impact on the service that they *aren't* monitoring. For example, the flight_info Probe reduces the response time of book_flight. These impacts occur due to monitors and services sharing the same, limited set of resources.

4.4 iTravel Requirements Analysis

The Requirements Analyser transforms the iTravel requirements iR (described in Section 4.2), into a set of utility functions iU[2] describing the utility to be gained for meeting these requirements. In iTravel's case, utility is directly and linearly computed from fines for non-compliance, however the set of utility functions may be modified or extended to add custom requirements or model some non-linear relationship. iTravel's requirements (section 4.2) are transformed into the set of iTravel utility functions, in format (service, quality type, min. quality level, min. sampling rate)↦utility:

(flight_info, Response Time, 0.25, 1) ↦ 0.165 (3300),
(book_flight, Response Time, 0.25, 1) ↦ 0.33 (6600),
(flight_info, Reliability, 0.95, 1) ↦ 0.5 (10,000),
(book_flight, Reliability, 0.98, 1) ↦ 0.5 (10,000),
(flight_info, Privacy, 1, 1) ↦ 1 (20,000),
(book_flight, Privacy, 1, 1) ↦ 1 (20,000),
(flight_info, Security, 1, 1) ↦ 0.25 (5,000),
(book_flight, Security, 1, 1) ↦ 0.5 (5,000).

This set describes the utility received from meeting each requirement. Utilities have been normalised to values between 0 and 1. Actual monetary values are shown in brackets, e.g. 1 (20,000) to clarify the derivation of the utility levels. All utility functions are defined incrementally, so that the total utility of a system is the sum of each individual utility function's value.

4.5 iTravel Optimization

The sets of available monitors, their impacts and their overheads were used to calculate the set of possible monitoring configurations with impacts for iTravel, $iMCI$. $iMCI$ maps each possible monitoring configuration to a set of net monitoring impacts. This information on monitoring requirements and impacts, along with the baseline response times and reliability for the iTravel web services has been used to perform optimization of iTravel's WS monitoring system.

The utility u of each configuration in $iMCI$ was calculated, and the configuration in $iMCI$ that yields the highest utility was selected and applied to the system. For iTravel, this configuration was the flight_info Interceptor and book_flight

[2] Available at http://www.ict.swin.edu.au/personal/gheward/

Interceptor both being set to monitor every quality of their respective target services, and every other monitor being disabled. The impact of this configuration is the sum of the impacts for each service (10% for Response Time on both services, 2% for reliability on flight_info, and 1% for reliability on book_flight). Therefore, the resulting response times of each service were $(2.5/(1 - 0.10) \approx 2.8)$, under no load, the reliability for flight_info was $(0.98 - 0.2 = 0.96)$, and the reliability for flight_info was $(0.99 - 0.01 = 0.98)$.

5 iTravel Optimization Evaluation

The iTravel case study has been implemented to verify our optimization technique and measure the possible QoS benefits of monitoring optimization. We now present a set of experiments and results to demonstrate the benefits of applying monitoring optimization to the iTravel system.

There always exists one or more optimal monitoring configurations (yielding the highest utility) in terms of monitoring coverage and QoS provided. These tests are designed to discover these optimal monitoring configurations in the iTravel system, and compare the utility and actual QoS of an optimal configuration to the utility and actual QoS of the corresponding maximum (un-optimized) monitoring configuration, in which all monitors run at sampling rates of 100%.

5.1 Experiment Configuration

The baseline QoS and monitoring impacts of the iTravel system were used to develop a simulation system that allows for the QoS of any particular monitoring configuration to be estimated through a series of simulated executions over a fixed period. For these tests, that fixed period was 6,000 seconds. This figure was selected through analysis of test results, since the results of all tests stabilised before 6,000 seconds. The simulations use response time measured through real service invocations to determine what the performance will be for a given monitoring configuration at any particular load level. Additionally, the reliability of all invocations is determined after each simulation.

As discussed above, a set of utility functions representing the iTravel requirements has been generated. These were used along with measured monitoring impacts to perform optimization on the iTravel monitoring system. The optimal monitoring configuration for iTravel was enabling both Interceptor monitors at 100% for all qualities of service, and disabling all other monitors.

Note that, since we are presenting a simple scenario, it is possible to select this optimal configuration manually by examining the system requirements and impacts. However, we intend for these optimizations to be applied to larger and more complex scenarios for which optimization would be too difficult to achieve without automation. Furthermore, optimization may be applied frequently at run-time, as parameters such as requirements and load levels change. In this case, it would not be effective to have a human constantly checking the system for changes and re-optimizing as they occur.

5.2 Results

We report the resulting average response times, reliability (percent of correctly returned invocations), and net utilities for each simulation. Response time and reliability results demonstrate that the performance and reliability of the iTravel system has increased by applying an optimal monitoring configuration. Utility results demonstrate that we have met iTravel's monitoring requirements. The total utility for the optimized configuration was higher than that of the maximum configuration in each test, as only the minimum valuable monitoring level was met. This allowed for a QoS increase, thus minimising the chance of a penalty to consumers for poor QoS.

Figure 3 shows response time versus load level for the iTravel scenario with unmonitored, maximum, and optimal monitoring configurations. The load levels on the horizontal axis represent the average number of active client requests, whilst the vertical axis gives the average response time over both services. The dotted line indicates average response times under maximum monitoring, the dashed line indicates the average response times under optimal monitoring, and the solid line represents the average response times under no monitoring. Since the response times for each service were almost identical, the results in Figure 3 represent the performance of both the book_flight and flight_info services.

Optimization reduced the average response time of the iTravel scenario by approximately 65% from maximum monitoring (from 33 to 12 seconds on average), and reduced the average response time impact of monitoring by over 50%(the optimal monitoring configuration was on average 11% slower than no monitoring, versus the maximum monitoring configuration's 68%).

The horizontal, solid line on Figure 3 shows the 10-second boundary, for which penalties apply in the iTravel system. The optimized configuration stays under this boundary for three times longer than the maximum monitoring configuration, i.e. the system with optimized monitoring dealt with three times as much load before a penalty for slow response times would have been paid.

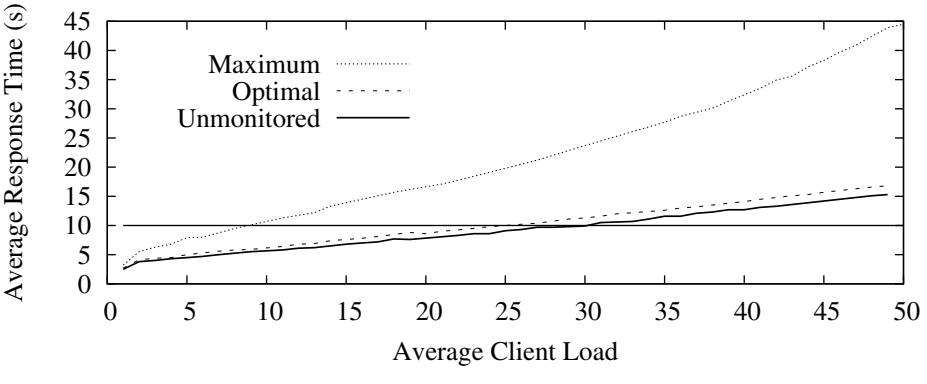


Fig. 3. iTravel Optimization Results

In terms of reliability, the unmonitored system maintained levels of 98% for flight_info and 99% for book_flight, whilst the optimal monitoring configuration yielded levels of 96% for flight_info and 98% for book_flight, and the maximum monitoring configuration yielded 92% for flight_info and 95% for book_flight.

The utility provided by the unmonitored solution was 0, since no requirements would have been able to be verified. Conversely, the maximum and optimal monitoring configurations both included complete monitoring coverage. Of the optimal and maximum monitoring configurations, only the optimal monitoring configuration yielded reliability levels that met requirements, so the utilities for reliability were received (0.5 for each service) under that configuration. For the optimal configuration, the utility at or below response time of 10 seconds was 4.245 (the highest achievable utility), and the utility above 10 seconds was 3.75. For the maximum configuration, the utility at or below response time of 10 seconds was 2.75, and the utility above 10 seconds response time was 2.25.

iTravel is a special case, since all of the utility functions have been directly derived from requirements with fines for non-compliance. For this reason, we can revert the expected utility values back to monetary terms, for comparison of each monitoring configuration. For this case study, 1 util is worth $20,000. Therefore, the expected benefits of the optimal configuration over the default, maximum configuration are $(4.245 - 2.75) \times \$20,000 = \$29,900$ when the response time is under 10 seconds, and $(3.75 - 2.25) \times \$10,000 = \$30,000$ when the response time is over 10 seconds.

The test results demonstrate that both the utility and QoS of the iTravel case study system were increased by optimizing the configuration of WS monitors. Although the results cannot be generalised to all WS systems as they depend on system properties, benefits can be substantial, even for a simple system. These benefits depend on the amount of overlapping functionality of a monitoring system, the level of impacts of monitors, the benefits of monitoring, and the benefits of achieving performance goals. As such, the benefit of optimization will increase as a WS system becomes more complex, with more overlapping monitors to choose from, more system elements to monitor, and a greater number of monitoring and performance requirements to meet.

6 Related Work

There are various methods for optimizing WS-based systems through selective execution or load balancing [4,5]. Whilst highlighting the need to optimize the performance of WS systems, none of these techniques relate to the management of a WS monitoring system.

Ranganathan and Dan present a Web Services management system to monitor and reallocate local system resources for services based on comparing their current QoS to their SLAs[6]. Whilst performing system-level administration, this method does not re-configure the web service monitoring system.

Baresi and Guinea present a method for dynamic monitoring of WS-BPEL processes, which uses high level monitoring rules [7]. These monitoring rules are used to control the monitoring of each WS-BPEL process. The rules are created with an equivalent of debug levels (one to five), which allows for optimization in terms of performance versus monitoring trade-offs at run-time. The monitoring level must be set for each service as a unit. Rather than assigning a monitoring resolution (debug level) to each individual service in the system, we assign a monitoring resolution to each quality of each service, directly reflecting requirements from SLAs. Furthermore, we enhance the optimization by selecting those monitors which will most efficiently monitor each service.

Overall, there have been numerous efforts for optimizing the QoS of web services and web service compositions, which consider the selection of services in order to optimize QoS. We have discovered no other work that optimizes a web services monitoring system by trading off between monitoring costs and benefits, directly translated from SLAs and other requirements for monitoring.

7 Conclusions and Future Work

We have described the optimization of a suite of WS monitors for a case study, and demonstrated the possible benefits of optimizing a WS monitoring configuration. This optimization uses sets of monitoring requirements and impacts of WS monitors to configure those monitors in a way that yields an optimal mix of monitoring coverage and monitoring impacts.

We will extend our optimization work in three directions. First, we will develop a heuristic approach to optimization to ensure scalability of the technique. This new heuristic technique will allow for faster optimization of a large system (hundreds of services and monitors), so that the system can be re-optimized either overnight, or continuously. Second, the framework and implementation will be extended so that run-time properties of the system being monitored are fed back into the monitoring optimization framework, for re-optimization at run-time. This will allow for the system to maintain an optimal configuration, even with changes to aspects such as requirements, system load, available monitors, or their impacts. Third, we will develop a method for using techniques to predict the future system state, and optimize a suite of monitors according to this predicted future state, rather than the current system state. This will include attempting to continuously verify or update the monitoring impacts on the system.

In the future, the complexity of this case study will be extended to allow for more services and monitors. Additionally, we will modify the case study to allow for run-time optimization and automated monitoring configuration, in order to demonstrate the possible benefits of these approaches.

Acknowledgements. This work is supported by the Australian Research Council in collaboration with CA Labs.

References

1. Heward, G., Müller, I., Han, J., Schneider, J.-G., Versteeg, S.: Assessing the performance impact of service monitoring. In: Australian Software Engineering Conference (ASWEC 2010), pp. 192–201 (2010)
2. Heward, G., Han, J., Müller, I., Schneider, J.G., Versteeg, S.: Optimizing the configuration of web service monitors. In: Maglio, P.P., Weske, M., Yang, J., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6470, pp. 587–595. Springer, Heidelberg (2010)
3. Heward, G., Han, J., Müller, I., Schneider, J.G., Versteeg, S.: Optimizing the configuration of web service monitoring configurations. Technical Report, Swinburne University of Technology (2010)
4. Ludwig, H., Dan, A., Kearney, R.: Cremona: An architecture and library for creation and monitoring of ws-agreements. In: International Conference on Service Oriented Computing (ICSOC 2004), pp. 65–74 (2004)
5. Herssens, C., Jureta, I.J., Faulkner, S.: Dealing with quality tradeoffs during service selection. In: International Conference on Autonomic Computing (ICAC 2008), pp. 77–86 (2008)
6. Ranganathan, K., Dan, A.: Proactive management of service instance pools for meeting service level agreements. In: Benatallah, B., Casati, F., Traverso, P. (eds.) ICSOC 2005. LNCS, vol. 3826, pp. 296–309. Springer, Heidelberg (2005)
7. Baresi, L., Guinea, S.: Towards dynamic monitoring of ws-bpel processes. In: Benatallah, B., Casati, F., Traverso, P. (eds.) ICSOC 2005. LNCS, vol. 3826, pp. 269–282. Springer, Heidelberg (2005)

Configuration Decision Making Using Simulation-Generated Data

Michael Smit and Eleni Stroulia*

Department of Computing Science
University of Alberta
Edmonton, AB, Canada
{msmit,stroulia}@cs.ualberta.ca

Abstract. As service-oriented systems grow larger and more complex, so does the challenge of configuring the underlying hardware infrastructure on which their consitituent services are deployed. With more configuration options (virtualized systems, cloud-based systems, *etc.*), the challenge grows more difficult. Configuring service-oriented systems involves balancing a competing set of priorities and choosing trade-offs to achieve a satisfactory state. To address this problem, we present a simulation-based methodology for supporting administrators in making these decisions by providing them with relevant information obtained using inexpensive simulation-generated data. Our services-aware simulation framework enables the generation of lengthy simulation traces of the system's behavior, characterized by a variety of performance metrics, under different configuration and load conditions. One can design a variety of experiments, tailored to answer specific system-configuration questions, such as, "what is the optimal distribution of services across multiple servers" for example. We relate a general methodology for assisting administrators in balancing trade-offs using our framework and we present results establishing benchmarks for the cost and performance improvements we can expect from run-time configuration adaptation for this application.

1 Introduction

Service-oriented architectures, cloud computing, Web 2.0, and 'smart' infrastructures are important technologies and areas of active software research today. Together, they promise to enable novel features of software systems, such as interoperability across organizations, cohesive integrated services that can be deployed and run quickly, and intuitive interactions across people and systems. These technologies are the current state-of-the-art; whether they are further developed or replaced, the functionalities they strive to provide will continue to be sought after. With these functionalities, however, come challenges inherent to complicated systems: there are multiple organizations involved, the behavior of the system is difficult to predict based solely on the behavior of the individual

* The authors acknowledge the generous support of iCORE, NSERC, and IBM.

E.M. Maximilien et al. (Eds): ICSOC 2010 Workshops, LNCS 6568, pp. 15–26, 2011.

components, and technical decisions may be dictated by business decisions made without consideration of technical challenges. With the many options involved (outsourced or self-owned? cloud computing or local infrastructure? green or cheap?), configuring applications built on these technologies is a challenge.

One common decision is the size, scale, and configuration of computing infrastructure. This task is complex even for technical experts. Decisions must be made about trade-offs between many configuration parameters and the overall cost and quality-of-service they imply. Configuration assistance, especially autonomic assistance, aims to support less-technical users in their decision making about systems with which they do not have expertise. Trade-offs of particular interest are those involving performance (quality-of-service) versus cost (financial).

We describe three novel contributions of our service-system configuration tool, and demonstrate each contribution using simulation-generated metrics[1]. First, question answering, where the simulation is run in specific scenarios designed to explore the system behavior under alternative conditions of interest. We demonstrate this method by answering one question about the parallelization of a service and another about the distribution of a service's operations over multiple servers (Section 4.1). Second, we describe our method for exploration of potentially conflicting configuration goals and trade-offs, where we use the data to help an administrator establish service-level agreements (SLAs), appropriate thresholds, accurate costing models, or appropriate configurations to meet a specific SLA or cost model (Section 4.2). Finally, we explore the effect of run-time configuration adaptation on the simulated application, determining the benefit of and targets for a future autonomic system. In particular, we explore two alternatives: one based on response time and server performance metrics, and another based on request statistics and composition knowledge (Section 4.3). The tool developed for this exploration could also be used to train administrators to adapt configurations to changing situations. Finally, we summarize the contributions of this paper in Section 5.

2 Background and Related Work

We consider a *service*, an application that offers a specific function, a "building block" for systems of services. A common software implementation specification is Web Services (WS), which offers a well-defined interface to server-side software and a suite of standards[2]. Services can be *composed* to provide more complex functionalities. *Service Level Agreements (SLAs)* can be used to govern these interactions. An SLA is a contract between two parties, where one promises a certain level of service and the other promises a corresponding payment for it.

[1] Metrics of any type will work as input to our implementation; the focus of this paper is not simulation but rather methods enabled by simulation data.

[2] The terms and language used throughout this document are those used for Web Services; this is done without loss of generality as the methodology is intended to be applicable to distributed applications generally, though the implemented tools may have dependencies based on WS technologies.

The level of service promised and expected can be described in technical or non-technical terms, however these terms will be typically measurable. SLAs can be general (promising 99% uptime) or negotiated based on needs/capabilities of the involved parties. Although SLAs are the current standard practice, they do not always ensure customer satisfaction. Blomberg [1] conducted a study of interactions between consumers and providers. She identified five problems with how SLAs were used in those interactions, including that SLA metrics do not always reflect the needs of the customer, that SLAs are not proactive, and that not enough information about the performance of the service is available to the client.

Decision making using simulation-generated data has been applied to domains including medicine [2], manufacturing [3], and software agents [4]. It commonly takes the form of answering specific questions (similar to our first contribution) but is less commonly used for configuration exploration and trade-off analysis or for run-time configuration adaptation (our second and third contributions).

Grundy *et al.* used a performance test-bed generator (MaramaMTE) [5] to prototype potential service compositions using simulated compositions, simulated load, and real-life services. Their methodology allows system architects to define a composition, test it, modify as required, and repeat. Their hybrid real-virtual approach assures accuracy but the number of tests that can be performed is limited; administrator intuition and understanding is required. Similarly, Chandrasekaran *et al.*[6] describe a tool for composing web services where known performance information is provided as input to a general simulation environment to simulate the orchestrated composition, producing a statistical estimate of the composed services' performance; however, they do not discuss any validation for their method, whether empirical or analytical.

Brebner *et al.* describe a tool that allows the user to define an SOA environment using building blocks like services, servers, workloads, and metrics based on UML diagrams [7] (instead of automatically as in our approach). Given parameters (*e.g.*, performance data and hardware), an event-based simulation produces performance metrics. This is part of a project looking at SOA performance [8,9].

Miller *et al.* [10] use simulation in combination with a workflow management tool to answer "what-if" questions about workflow adaptations and their effect on quality of service. They do not model the individual nodes in the workflow and they assume QoS information is already available for individual nodes. They describe a hypothetical case study but do not offer a validated implementation.

Almeida and Menasce [11] offer a general method for capacity planning in client-server systems, making explicit the need for models of workloads that the architecture is expected to support and the way its performance may change when aspects of the workload change. This approach is not service-focused and does not use simulation to produce data.

3 Simulation-Generated Dataset

In this section, we describe the generation of the datasets used by our reasoning tools. For a more detailed discussion, the reader should review prior work [12,13].

The simulation framework is a suite of tools that enable (a) the creation of a model of a service-oriented application, and (b) the "virtual" execution of the model to generate and record data about its run-time behavior.

The framework has four main components. The *simulation engine* provides an environment for running simulations (clock management, networks, basic service functionality, composition, *etc.*). *wsdl2sim* takes as input a service interface (specified as a WSDL) and its performance model, and generates a simulation to be run by the engine. The performance model of a service is constructed automatically using load testing[3] under various configurations and fitting a curve to the resulting metrics. The generated simulations are discrete-event and run in simulated time (faster than real time). *JIMMIE* uses an XML-based language to systematically re-configure the simulation and repeatedly run it to test different scenarios. The *dashboard* module facilitates collecting, storing, and visualizing metrics generated by the simulation, as well as interacting with the simulation.

We used this framework to simulate TAPoRware, a text-analysis system that provides a public web services interface. Its focus is on data processing (CPU-bound), with the movement of data being a secondary but important concern. TAPoRware is a single web service with 44 operations, implemented in Ruby. Each operation runs in $O(n)$, bounded by CPU time relative to the size of the input. The tools covered in this paper are listing the words with their counts, generating word clouds, finding the use of a word in context (concordance), and finding two words located near each other in text (co-occurrence).

We evolved the simulated model to add features not in the original application. We added metrics capturing code to generate the dashboard, created a load balancer, added a request generator (Section 3.1), extended the configuration files to allow for JIMMIE integration, and added the ability to manually re-configure at run-time. We treated the operations as if they belonged to separate services.

3.1 Generating Performance Metrics through Simulation

The system under examination is simulated under varying configurations (systematically and repeatably) to predict its behavior and performance in different conditions. The data produced by the simulation is determined by two factors: (a) input (incoming requests and their parameters) and (b) the configuration of the simulated application. The generated data is periodic snapshots of performance data under different combinations of these two factors.

Incoming requests vary based on several parameters: the arrival rate of requests, the type of each request (*i.e.*, the operation being invoked, as each has a different performance profile), and the size of each request. The simulation framework offers the ability to write custom request generators; we authored four: a) stress testing; b) reading from log files (either from the real-world application or one of the other generators); c) stochastic, based on probability distributions for request arrival rate (Poisson), type of request (weighted random based on real logs), and size of request (parameterized double gaussian and exponential distributions based on curve-fitting size distributions from the real log files); d)

[3] soapUI (`http://www.soapUI.org`) was used to generate SOAP messages.

stochastic, where the distribution parameters can be modified at run-time. Each of these request generators records repeatable traces of generated traffic.

The configuration of the simulated application is described with an XML document "understood" by the simulation framework. The specification document identifies the available servers, the software services of the system and their distribution on these servers, the appropriate "stub" classes that should be loaded to emulate them, and the request generation strategy and parameters. Note that the JIMMIE component of the simulation framework can systematically produce a multitude of such configuration specifications in order to run automatically hundreds of simulated experiments.

Given such a specification document, our framework can be invoked to simulate the subject system and record the performance metrics. These metrics are captured periodically, persisted to database storage, and (optionally) visualized at run-time using the dashboard. Metrics captured include queue length for each server, CPU utilization for each server and globally, response time for each request and rolling averages, the total time required to process the sample set of requests, and other related metrics based on those commonly used in SLAs. Metrics are stored every 5 seconds; simulation-wide averages are stored at the end of the simulation period.

4 Reasoning Using Simulation-Generated Data

This section describes three methodologies and tools we have developed to reason about specific questions around the system configuration, and tested using simulation-generated data[4]. First, answering questions about the system's behavior under specific configuration decisions. Second, exploring general trade-off analysis with conflicting goals, such as cost and performance. Finally, testing how simulation data can be used at run-time to identify system reconfigurations.

4.1 Answering Questions About Configurations

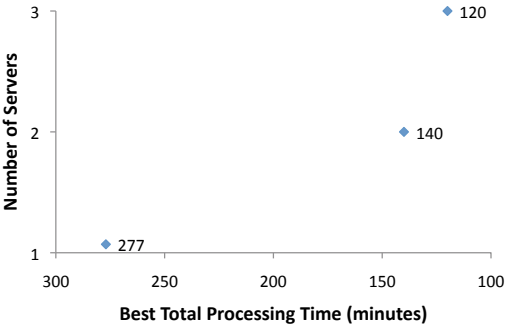
The first use of simulation-generated data is to answer specific questions around the system's behavior in different configuration and load conditions. In this approach, the configurations are systematically modified to test a series of configuration options, and the results are compared to answer the question of interest, usually around making a trade-off. To illustrate this scenario, consider two questions based on a real-world TAPoRware issue, namely the benefits of concurrency and what distribution of services on what number of servers is best suited to meet expected load.

First, *for a single-server installation of TAPoRware, does allowing the List Words operation to process multiple responses simultaneously improve the overall response time?* JIMMIE was used to generate configuration files specifying 1000 requests of the List Words type, size 1000KB, but with requests arriving with

[4] These tools are intended to be general; other performance metrics from other applications could be used in lieu of the sample simulated application used here.

Svc.	Req.	Response Time
1	1	29.81 ± 0.48
2	1	29.83 ± 0.53
1	2	58.88 ± 4.1
2	2	58.76 ± 1.0
1	5	142.92 ± 22.3
2	5	145.77 ± 32.3

(a) Response time for 50 1MB requests with varying service and request concurrency.



(b) The best total time for 1, 2, and 3 servers with varying operation distributions.

Fig. 1. Question answering results

various levels of concurrency: 1, 2, or 5. A single-server single-operation server was simulated and configured to process 1 or 2 requests concurrently (with any outstanding requests waiting in a queue). The experiment outcomes are shown in Figure 1a. The results show that serving 2 concurrent requests instead of one does not improve total processing time or response time by a significant or reliable amount. This is not surprising given the performance model; a single request is sufficient to occupy the only available processor. The question could be extended to ask what gains result from adding a second processor to a concurrent service.

The second question considers the *preferable distribution of 6 services over 2 or 3 servers*. The goal is to identify the configuration that camn potentially produce the best response times (performance) versus lowest cost trade-off.

Before using our simulation-based process, one of the authors of this paper used the same information available to this tool and designed his own configuration, which he expected to be the best cost-perfromance compromise. We then used JIMMIE to generate all possible distributions of 6 services on 2 and 3 servers[5]. Traffic was generated based on logs from the existing service. The simulation was run faster than real time, so each configuration was simulated in between 2 and 4 minutes. One server hosting all 6 services is used as a baseline.

The best total times for 1, 2, and 3 servers are shown in Figure 1b. The fastest two-server configurations took 140 minutes to process all of the incoming requests, *i.e.*, twice the cost, half the time. The fastest three-server configurations completed shortly after the traffic generator stopped generating requests (120 minutes), 57% less time. This represents 3 times the cost for a 2.3 performance improvement factor: a more expensive improvement for the expected load. Service distribution is important: 75% of the three-server configurations performed worse than the best two-server configuration.

[5] The idea of using different distributions instead of a complete mirror of the service allows faster operations to be assigned to one server and slower operations to another; a fast operation stuck behind a slow operation in a queue will experience a more dramatic increase in its response time.

The manually-designed configuration took 137 minutes to run, 14% longer than the best three-server configurations and 2% faster than two servers. This confirmed our intuition that configurations produced by experts based on their understanding of the software, expected load, and environment may not necessarily meet the properties desired of the system.

4.2 Trade-Off Analysis and SLA Decision Support

Configuring a software system involves meeting a set of goals, which might be well-specified or only intuitively understood, and might be complementary or might conflict. Our trade-off analysis methodology supports decisions involving the balancing of conflicting goals, whereby one might relax goals in one area in order to achieve other goals, perceived as more important. Without loss of generality, we focus our discussion on cost and performance metrics, as they are common and more readily quantifiable. One of the shortcomings of SLAs identified by Blomberg [1] was that the parties involved in creating SLAs lacked the information and intuition needed to make appropriate decisions about quality of service levels; our approach attempts to provide both information and intuition when planning.

In the previous section, a specific trade-off question was answered. This is effective if the system manager knows which few configuration options should be considered as alternatives. When the right set of options is not known, and the goal is a general exploration of the trade-offs, a large number of possible configurations can be simulated to generate a knowledge base that powers a decision support tool. Using this knowledge base, we visualize various conflicting or correlated metrics to understand the impact that one goal will have on another. The system manager can select potentially viable configurations using a visual tool. Based on their selections, we develop a "fitness score" for each configuration.

Visual Selection Tool. The configuration exploration simulations are visualized as a series of weighted scatter plots in a two-dimensional space, where the two axes correspond to two metrics, offering a head-to-head comparison. For example, Figure 2 shows a sample plot of correlated values, the average response time (x) versus the total time required to complete the processing. The configurations are clustered based on their values for the two metrics. Each point on the plot corresponds to a cluster, and is sized based on the number of configurations that belong in this cluster. The user is shown a series of these plots and is asked to draw a rectangle around the configurations that meet the requirements of the system under configuration. Each successive plot would have clusters colored different shades based on how many configurations in that cluster have been previously selected. Once the user has made all of their selections, he is presented with a list of candidate configurations based on how often those configurations were selected on individual plots. To select a final configuration, the user can watch a recording of the simulation for each candidate to observe metrics in action, choose based on a simple metric (such as the cheapest, the most energy efficient...), or trust a more complex fitness score (as explained in the next section). Hovering over a point produces a "tool-tip" listing the configurations involved, important metrics, and key configuration parameters.

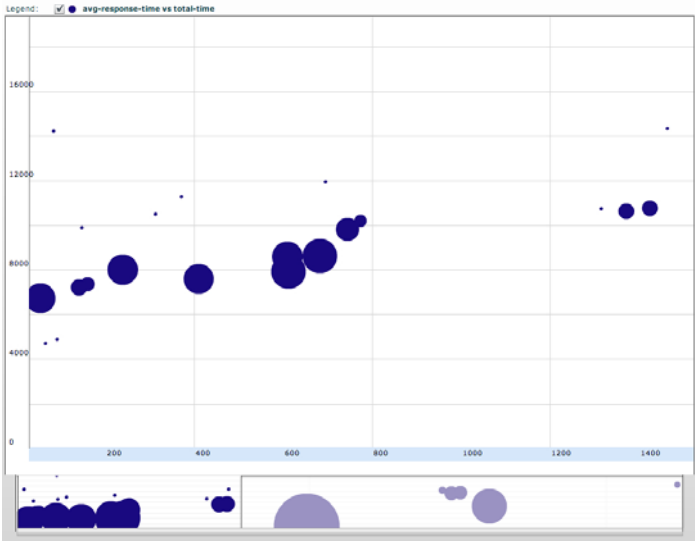


Fig. 2. A visualization of configurations' Average Response Time versus their overall time to process

Fitness Score. Based on the user's selected simulations, the visual tool computes a fitness score. Each configuration, in each head-to-head comparison, is given a comparison score h_{ij} based on its values for the compared metrics i and j:

$$h_{ij} = weight_{ij}[i] * value_{ij}[i] + weight_{ij}[j] * value_{ij}[j]$$

The *weight* is determined by two factors. First, how many configurations were selected out of the total set: if the user accepted a metric's value for most configurations, that metric is relatively unimportant in distinguishing between good and bad configurations. Second, how much of the total range of that metric is selected. If the user was more selective, the metric is more important. For example, consider a comparison of average response time (x) and cost (y) that plots 100 configurations, where response time ranges from 1 to 101 seconds and cost from 200 to 500 dollars. The user draws a selection box from (1,200) to (10,400); there are 34 configurations with response time between 1 and 10, and 56 configurations with cost between 200 and 400. The first factor is $1 - \frac{34}{100} = .66$ for response time and $1 - \frac{56}{100} = .44$ for cost. The second factor for response time is $1 - \frac{(10-1)}{101-1} = .91$ and for cost is $1 - \frac{400-200}{500-200} = .33$. Thus $weight_{ij}[r.t.] = 1.57)$ and $weight_{ij}[cost] = .77$. Note that response time and cost may have different weights when compared to other metrics.

The $value_{ij}[k]$ is normalized based on the relative position of the configuration's value for metric k: at the "good" end of the selection is 2, at the "bad" end is 1, everything in between on a linear scale. Values outside the selection have value 0. Returning to our example, a configuration with response time of 1 (best in the selection) and cost of 500 (outside the selection) would have values 2 and

0, respectively. The fitness score for that configuration for this pair of metrics would be $h_{ij} = 1.57 \times 2 + .77 \times 0 = 3.14$ out of a possible 8 points.

Demonstration. We used the configuration traces from Section 4.1, namely a performance versus cost trade-off for approximately 800 candidate configurations. Using the visual selection tool, we selected ranges in head-to-head comparisons based on the perceived desires of the user community (fast response, consistently, at low cost). Comparisons deemed unimportant were ignored (value = 0). There were 6 metrics, and $\binom{6}{2} = 15$ potential head-to-head comparisons; only 6 comparisons were considered meaningful. Using the visual selection tool, 153 candidate configurations were identified, each having been selected 4 times.

This reduced the candidate configurations by 81%. There remained variation within the remaining configurations; the fitness score allows us to rank the configurations using preferences expressed in the visual tool. We computed the fitness score for each of the six comparisons. Of the 153 candidate configurations identified by the visual tool, 9 were missing from the top 153 of the new ranked list, in all cases due to being very close to the boundaries of the selection box (*i.e.* marginal candidates). This identified 49 top configurations (all within .1 of 41.2, with the remaining configurations under 40, on a theoretical maximum score of 48). This is a 94% reduction. In the future, we plan to empirically validate whether this selection process is actually satisfactory to the user.

4.3 Run-Time Configuration Changes

Configuring software prior to deployment involves identifying a configuration to meet projected future needs. These may not be as predicted or may change over time, and configurations may not perform as expected. Changes will be necessary, and can be made manually or autonomically. Manual changes require well-trained experts who are able to make appropriate configuration changes in response to changed circumstances. We propose and implement a simulation-based tool to train administrators in this task. Providing metric visualization and capturing tools allows for interactive and visual training. Combining modifiable request generation with run-time configuration adaptation through a GUI in simulation allows for training in varying circumstances. It can also be used to train an autonomic system capable of changing system configurations in real-time.

Here we examine the potential benefit of (substantial and expensive) adding autonomic managers to the software by conducting the following experiment. A colleague unassociated with this project generated 160 minutes of variable

Table 1. Cost and performance metrics (and improvements) for 5 configurations

Config	Response Time	Total Time	Cost Increase	Perf. Increase
3-server	1467.3	14272.5	0.0%	0.0%
4-server	698.8	11887	11.0%	52.4%
5-server	316.5	10433.5	21.8%	78.4%
6-server	151	9826.5	37.7%	89.7%
Variable	375	11211.5	3.1%	74.4%

Table 2. Cost and performance metrics (and improvements) comparing manual configuration, manual configuration with composition, and static configurations

Config	Response Time	Total Time	Cost Increase	Perf. Increase
3-server	1353.8	12555	0.0%	0.0%
4-server	611.2	10801	14.7%	54.9%
5-server	199	9775.5	29.8%	85.3%
6-server	52.9	9662.5	53.9%	96.1%
Manual	149.9	9847	-3.5%	88.9%
w/ Composition	91.38	9763	11.0%	93.3%

request traffic, which we logged and used as a test set. Five application configurations were tested: three, four, five, and six servers, and one where an expert user added and removed 1-6 servers at run-time. The distribution of operations was not changed. The expert user had no prior knowledge of the contents of the recorded requests; his decisions were based on watching performance metrics: the load on each server, the queue at each server, and the overall response time. For each configuration, we used average response time and total time as performance measures. We also calculated configuration cost, based on total active CPU time (loosely, a cloud-computing scenario[6]). A server contributed to the cost if it was configured to process requests, whether it actually received requests or not.

The full results are in Table 1; we set a fixed three-server configuration as the baseline and made comparisons relative to that. Fixed five- and six-server configurations performed better than the variable configuration, but were more expensive. Compared to a 3-server configuration, the manually varied configuration was 3% more expensive but performed 75% better. Autonomic changes could allow for faster, more fine-tuned changes; the current method was limited by human response times to a simulation running 100 times faster than in real life. On the other hand, autonomic changes might not be as "intelligent" as manual changes. Nonetheless, based on these results we believe an autonomic manager could improve performance while reducing costs.

Composition-Aware Adaptation. We also evaluated the potential of using composition information to better inform run-time configuration adaptation. We simulated a scenario based on a real-world usage of TAPoR: pre-indexed texts which can be analyzed, in addition to user-submitted texts. A web service request for a list of available indexes precedes some interactions with the service. Some (but not all) interactions with the service follow pre-composed "recipes", where a specific sequence of operations is called in rapid sequence. We modeled the operations as individual services to allow the recipes to represent composed services.

To test this approach, we modified a random 70% of the requests in the data set used above, adding composition. We augmented the user display with a request-type arrival rate metric. The user was able to see the distribution of

[6] The estimated costs depend on the price model assumed; in this work, we assume a hardware-leasing model. Given alternative cost-calculation models, the estimated costs would be different. Our approach is independent of any particular cost models.

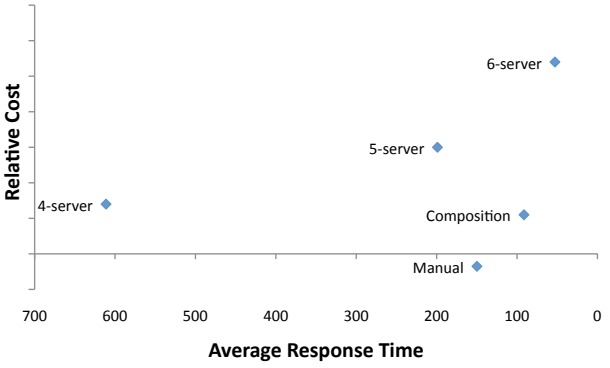


Fig. 3. Relative cost and performance for manual configuration, manual configuration with composition, and static configurations

arriving requests and predict future capacity needs. For example, a request for the list of available indices takes little time and will not impact typical performance metrics, but is a reliable predictor of future requests. This experiment configuration does not perfectly simulate the real-world scenario but we expect it to provide a fairly accurate indication as to the viability of using composition-related knowledge for this application. For this experiment, the administrator used two modification strategies. The first was based on using performance metrics as above, regardless of composition. The second used request metrics: the number and type of requests sent, and the number of requests that have not received a response (as tracked by the traffic generator).

The full results are in Table 2; the trade-offs are shown in Figure 3. When using composition knowledge, we achieved a 93% reduction in response time for an 11% cost increase. Without composition, we achieved a slightly worse reduction in response time (89%) but with a 3.5% cost *decrease* compared to the base. Based on this, admittedly simple, experiment, we found no evidence that including composition in our decision-making improved the run-time adaptation.

5 Conclusion

The work discussed in this paper simulated a real-world application using a services-aware simulation framework, which we used to generate performance data in a variety of configurations. We used this simulation-generated data to present three contributions to configuration planning. First, a question-answering methodology and tool, which we demonstrated by answering two questions. Second, a trade-off analysis decision support tool to help non-technical users derive increased value from their services by improving their understanding of the software service system. We described how this visual tool can be used by users to identify their preferences, which we use to assign a fitness score to future simulation results. Third, we describe a potential training/evaluation tool, but more importantly we use this tool to assess the viability of an autonomic manager for this particular application, showing that we can gain substantial performance

increases for slightly increased cost by manually provisioning just-enough resources. We also suggest that for this service system, knowledge of how services are composed is not likely to improve an autonomic manager.

This work is based on simulated data, and though the simulation when compared to real-world metrics in a set of configurations was statistically identical, the simulation cannot be guaranteed to be accurate. It will also not account for pathological cases. In the near future, we will be using simulation-generated data to inform an autonomic management system capable of re-configuring a system at run-time based on correlating observations of the current environment with past experience. Additional efforts focus on using simulation-generated data to reason more generally about value and perceived quality in trade-off analysis.

References

1. Blomberg, J.: Negotiating meaning of shared information in service system encounters. European Management Journal 26(4), 213–222 (2008)
2. Groothuis, S., Godefridus, van Merode, G., Hasman, A.: Simulation as decision tool for capacity planning. Computer Methods and Programs in Biomedicine 66(2), 139–151 (2001)
3. Uribe, A.M., Cochran, J.K., Shunk, D.L.: Two-stage simulation optimization for agile manufacturing capacity planning. International Journal of Production Research 41(6), 1181–1197 (2003)
4. Kuehne, R., Wille, C., Dumke, R.: Software agents using simulation for decision-making. SIGSOFT Softw. Eng. Notes 30(1), 5 (2005)
5. Grundy, J., Hosking, J., Li, L., Liu, N.: Performance engineering of service compositions. In: Proceedings of IW-SOSWE 2006, pp. 26–32. ACM, New York (2006)
6. Chandrasekaran, S., Miller, J., Silver, G., Arpinar, B., Sheth, A.: Performance analysis and simulation of composite web services. Electronic Markets 13(2), 120–132 (2003)
7. Brebner, P.C.: Performance modeling for service oriented architectures. In: Proceedings of ICSE 2008, pp. 953–954. ACM, New York (2008)
8. Brebner, P., O'Brien, L., Gray, J.: Performance modeling for e-government service oriented architectures (SOAs). In: Ashley Aitken, S.R. (ed.) ASWEC, Australia, pp. 130–138. ACS (March 2008)
9. O'Brien, L., Brebner, P., Gray, J.: Business transformation to SOA: aspects of the migration and performance and QoS issues. In: Proceedings of SDSOA 2008, pp. 35–40. ACM, New York (2008)
10. Miller, J.A., Cardoso, J., Silver, G.: Using simulation to facilitate effective workflow adaptation. In: Annual Simulation Symposium, p. 0177 (2002)
11. Menasce, D., Almeida, V.: Capacity Planning for Web Services: metrics, models, and methods. Prentice Hall PTR, Upper Saddle River (2001)
12. Smit, M., Nisbet, A., Stroulia, E., Edgar, A., Iszlai, G., Litoiu, M.: Capacity planning for service-oriented architectures. In: Proceedings of CASCON 2008, pp. 144–156. ACM, New York (2008)
13. Smit, M., Nisbet, A., Stroulia, E., Iszlai, G., Edgar, A.: Toward a simulation-generated knowledge base of service performance. In: Proceedings of MW4SOC 2009, Newport Beach, CA, USA (2009)

On the Formal Specification of Regulatory Compliance: A Comparative Analysis

Amal Elgammal*, Oktay Turetken,
Willem-Jan van den Heuvel, and Mike Papazoglou

European Research Institute in Service Science (ERISS), Tilburg University,
Tilburg, The Netherlands
{a.f.s.a.elgammal,o.turetken,w.j.a.m.vdnheuvel,
m.p.papazoglou}@uvt.nl

Abstract. Today's business environment demands a high rate of compliance of service-enabled business processes with which enterprises are required to cope. Thus, a comprehensive compliance management framework is required such that compliance management must crosscut all the stages of the complete business process lifecycle, starting from the very early stages of business process design. Formalizing compliance requirements based on a formal foundation of an expressive logical language enables the application of associated verification and analysis tools to ensure the compliance. In this paper, we have conducted a comparative analysis between three languages that can be used as the formal foundation of business process compliance requirements, focusing on *design-time* phase. Two main families of languages have been identified, which are: the temporal and deontic families of logic. In particular, we have considered LTL, CTL and FCL. The comparative analysis is based on the capabilities and limitations of each language and a set of required identified features.

Keywords: Compliance requirements specifications, linear temporal logic, Regulatory compliance, computational tree logic, formal contract language.

1 Introduction

Today's business climate demands a high rate of compliance of business processes with which Information Technology (IT)-minded organizations are required to cope. Compliance regulations, such as Basel II, Sarbanes-Oxley and others require all organizations to review their business processes and service-enabled applications, and ensure that they meet the compliance constraints set so forth in the legislation.

Compliance is mainly ensuring that business processes, operations and practices are in accordance with a prescribed and/or agreed on set of norms [1]. A compliance constraint (requirement) is any explicitly stated rule or regulation that prescribes any aspect of an internal or cross-organizational business process.

* This work is a part of the research project "COMPAS: Compliance-driven Models, Languages and Architectures for Services", which is funded by the European commission, funding reference FP7-215175.

E.M. Maximilien et al. (Eds) : ICSOC 2010 Workshops, LNCS 6568, pp. 27–38, 2011.

SOA is an integration framework for connecting loosely coupled software modules into on-demand business processes. BPs form the foundation for SOAs and require that multiple steps occur between physically independent yet logically dependent software services [2]. The control and disclosure requirements originating from multiple compliance sources create auditing demands for SOAs.

One of the key requirements of a generic compliance management approach is that it should be sustainable throughout the business process (BP) lifecycle [1]. Compliance management should be considered from the early stages of business process design, thus achieving compliance-by-design, which must be further integrated with dynamic monitoring of the running instances. The emphasis in this paper is on requirements applicable to design-time phase of the BP lifecycle, while the discussions on the requirements applicable to later phases (runtime and offline phases) are kept limited.

Founding the specification of business process compliance requirements on a formal logical language enables their automatic verification and analysis against business process specifications. However, the complexity of the formal language must not become an obstacle for the specification and for their validation. It is important to find an appropriate balance between expressiveness, formal foundation, and potential analysis methods [3].

In this paper, we conducted a comparative analysis between three languages that can be used as the basic building blocks of a comprehensive *Compliance Request Language* (*CRL*) for the formal specification of compliance requirements, focusing *primarily* on *design-time* verification. In particular, we consider two families of logics that have been used successfully in the literature, namely deontic and temporal families of logic. More specifically, we consider Formal Contract Language (FCL) [4] from the deontic logic family. On the other hand, we consider Linear Temporal Logic (LTL) and Computational Tree Logic (CTL) from the temporal logic family [5-6]. We applied these languages on the specification of a wide range of compliance requirements of two industrial case studies explored within the EU funded COMPAS research project [7]. Then a comparative analysis was conducted based on the capabilities and limitations of each language against a set of identified key features. The comparative analysis reflected that the decision on the selection of a particular formal language is context-dependent involving various factors including the nature, complexity and source of compliance requirements. However, based on the results of the comparative analysis, we can argue that the temporal logic has advantages over others with regard to the specification of regulatory constraints.

The rest of this paper is organized as follows: Section 2 highlights the key features that should be maintained by a comprehensive compliance request language. Section 3 presents a simplified motivating scenario used as the running example throughout this paper. Section 4 briefly describes the key concepts and rules of the formalisms analyzed and examine their capabilities to express compliance requirements from the running scenario. The comparative analysis is drawn in Section 5. Related work is summarized in Section 6. Finally, conclusions and ongoing work are highlighted in Section 7.

2 Required Features of a Compliance Request Language

In order to reveal the features that should be possessed by a language to be used for the formal specification of compliance requirements, we have analyzed [8] a wide range of compliance legislations and relevant frameworks such as Basel II, Sarbanes-Oxley, IFRS, FINRA (NASD/SEC), COSO, COBIT and OCEG. This set of regulations and frameworks constituted a faithful representation of the range of compliance requirements that can be found within compliance legislations. We also conducted case studies on industry processes [9] that are subject to various regulatory compliance requirements (also discussed in Section 3). Based on the findings, we have identified a set of features that should exist in CRL. These features can be summarized as follows:

- *Formality*: The CRL should be formal to pave the way for the application of associated automatic analysis, reasoning and verification tools and techniques.
- *Expressiveness*: The CRL should be expressive enough to be able to capture the intricate semantics of compliance requirements.
- *Usability*: The CRL should not be excessively complex to inhibit users to understand and use it.
- *Consistency checks*: Contradictions and conflicts might arise between compliance requirements particularly when they originate from different sources. It is desirable for the CRL to provide mechanisms to identify and resolve these inconsistencies.
- *Normalization*: This feature refers to cleaning-up of the requirements specification to identify and remove redundancies and to make implicit requirements explicit.
- *Declarativiness*: Compliance requirements are commonly normative and descriptive, indicating what needs to be done [1]. Therefore, declarative languages are more suited to their formal representation.
- *Generic:* Compliance requirements can be constraints on the control-flow (sequence and timing of activities), data (data validation and requirements), and resource perspectives (task allocation and data access rights). The CRL should enable the specification of the requirements regarding to these perspectives.
- *Symmetricity*: This feature refers to the ability to annotate business process models with compliance requirements. The annotation helps users to understand the interplay between the two specifications.
- *Non-monotonicity:* A violation to a compliance rule is not necessarily an error. Non-monotonic rules are open to violation to a certain extend and under specific conditions. Depending on the rigidity of the rule, the process expert can decide on the type of the rule, and the exceptions under which a specific rule can be overridden and the priorities among them.
- *Intelligible feedback:* Indicating whether there is a violation to a specific rule is not sufficient. It is important to provide the user with guidance of why a violation occurs and how to resolve compliance deviations.
- *Real-time support:* The language should be able to express real-time requirements, which are likely to appear in compliance sources. For example: Activity *A* should occur within time period *k*.

3 Running Scenario

The loan approval scenario is one of the industry case scenarios explored within the EU funded COMPAS research project [9]. The general environment in which this particular scenario takes place is banking e-business applications. Taking into account the demands for strong regulation compliance schemes, such as Basel II, Sarbanes-Oxley (SOX), IS0 27000 and sometimes contradictory needs of the different stakeholders, such scenarios raise several interesting compliance requirements.

Table 1. An excerpt of compliance requirements relevant to the case scenario

ID	Compliance Requirement (Context/Process Specific Interpretation)	Compliance Source
R1	*Only Post-processing Clerk and Supervisor roles can access the "Credit Bureau service".*	- Internal Policy
R2	*Customer bank privilege check is segregated from credit worthiness check.*	- SOX Sec.404 - ISO 27002 - 10.1.3
R3	*If the loan request's credit exceeds 1 million EURO the Clerk Supervisor checks the credit worthiness of the customer. The lack of the supervisor check immediately creates a suspense file. In case of failure of the creation of a suspense file, the manager is notified by the system and Post-processing Clerk is allowed to do the check.*	- Internal Policy - SOX Sec.404
R4	*As a final control, the branch office Manager has to check whether the request is profitable and risks are acceptable before making the final approval.*	- SOX Sec.404.
R5	*If loan conditions are satisfied, the customer can check the status of her loan request infinitely often until the customer is notified.*	- Internal Policy
R6	*If credit worthiness check activity is performed, then there exists an activity 'evaluation of the loan risk' that should be performed by the manager.*	- Internal Policy
R7	*The Credit Broker can start a loan (approved by the customer), only if 5 workdays or more have elapsed since the loan approval form was sent.*	- SOX Sec.404

The brief flow of the process is as follows: Once a customer loan request is received, the *credit broker* checks if customer's banking privileges are suspended. If privileges are not suspended, the credit broker accesses the customer information and checks if all loan conditions are satisfied. Next, a loan threshold is calculated, and if the threshold amount is less than 1M Euros, the *post processing clerk* checks the credit worthiness of the customer by conducting the credit bureau service. Next, the *post processing clerk* initializes the form and approves the loan. If the threshold amount is greater than 1M Euros, the *clerk supervisor* is responsible for performing the same activities instead of the post processing clerk. Next, the manager evaluates the loan risk, after which she normally signs the loan form and sends the form to the customer to sign. Table 1 lists an excerpt of the compliance requirements that are relevant to the scenario. The first and second columns of the table give a unique ID and a brief description of the original compliance requirement as they are in sources, respectively. Third column gives case scenario specific interpretation of the compliance requirement (internalized compliance requirements). Finally, the fourth column refers to the associated compliance source(s) (e.g. a legislation document).

4 Formalisms under Consideration

The following sub-sections present a brief description of the basic concepts and rules (syntax) of the considered logical languages, and examine their applicability to represent compliance requirements of the running scenario as described in Table 1.

4.1 Linear Temporal Logic (LTL)

LTL [5], [6] is a logic used to formally specify temporal properties of software or hardware designs. In LTL, each state has one possible future and can be represented using linear state sequences, which corresponds to describing the behavior of a single execution of a system. The formulas in LTL take the form Af, where A is the universal path quantifier and f is a path formula. A path formula must contain only atomic propositions as its state sub-formulas. The formation rules of LTL formulas are as follows:

- ⊤ and ⊥ are formulas (⊤ represents tautology and ⊥ represents contradiction).
- If $P \in AP$, where AP is a non-empty set of atomic propositions, then P is a path formula.
- If f and g are path formulas, then $\neg f$, $f \vee g$, $f \wedge g$, Xf, Ff, Gf, $f\ U\ g$, $f\ W\ g$ are path formulas ('∨' represents 'or' and 'Λ' represents 'and' operators):
  - *G (always)* indicates that formula *f* must be true in all the states of the path.
  - *X (next time)* indicates that the formula *f* is true in the second state of the path.
  - *F (eventually)* indicates that formula *f* will be true at future state of the path.
  - *U (until)* indicates that if at some future state the second formula g will be true, then, the formula *f* must be true in all the subsequent states within the path.
  - *W (weak until)* represents the same semantics as *until*, however it is evaluated to true even if the second formula g never occurs (note that $pWq \equiv G(p) \vee (p\ U\ q)$).

4.2 Computational Tree Logic (CTL)

CTL [5], [6] is also a logic used to formally specifying temporal properties of software or hardware designs. CTL differs from LTL in terms of their underlying model of time. As opposed to LTL, in *branching* temporal logics, each moment in time may split into various possible futures. Hence, the structure under which branching temporal logic formulas are interpreted is represented as infinite computational tree, which describes the behavior of the possible computations of a nondeterministic program [10]. A well-formed CTL formula over a set of atomic propositions $AP = \{p, q, ...\}$ (where A is the universal path quantifier) can be formed as follows (in BNF notations):

- $\varphi ::= \top \mid \bot \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi \mid AX\varphi \mid EX\varphi \mid AF\varphi \mid EF\varphi \mid AG\varphi \mid EG\varphi \mid A[\varphi U \psi] \mid E[\varphi U \psi] \mid A[\varphi W \psi] \mid E[\varphi W \psi]$, such that:
  - ⊤ represents tautology and ⊥ represents contradiction symbols.
  - *G*, *F*, *X*, *U, W* are the temporal operators 'always', 'eventually', 'next', 'until' and 'weak until' as defined in Section 4.1.

- *A* and *E* stand for the universal (for *All* paths) and existential (there *Exists* a path) quantifies, respectively.
- Each CTL operator should be a pair of symbols. The first symbol is a quantifier (*A* or *E*), and the second symbol is a temporal operator.

4.3 Formal Contract Language

FCL is a combination of an efficient non-monotonic formalism (defeasible logic) and deontic logic of violations. The FCL language consists of two sets of atomic symbols: A finite set of literals (propositions) that represent state variables, and a finite set of events. The logical operators that are supported are as follows: (i) ';' the sequence operator, (ii) 'Λ' conjunction operator and (iii) '∨' disjunction operator. A rule in FCL is an expression of the form: $r: A_{1,}A_2, \dots, A_n \vdash B$, where *r* is the identification of the rule, $A_{1,}A_2, \dots, A_n$ is the set of premises (propositions) and *B* is the conclusion of the rule. The rule is built from a finite set of atomic propositions, logical operators, and a set of deontic operators, which are; (i) Negation (¬), (ii) Obligation (*O*), (iii) Permission (*P*), and (iv) the Contrary to duty operator (⊗ or CTD). Contrary to duty operator is used to specify the violations and the obligations arise as a response to the violations. The rules are formed as follows:

- Each atomic proposition is a proposition.
- If *P* is an atomic proposition, then $\neg P$ is a proposition
- If *P* is an atomic proposition, then *OP* is an obligation proposition, *PP* is a permission proposition. Obligations and permission propositions are deontic propositions.
- If $P \vdash P_1 \otimes P_2 \otimes \dots P_n \otimes \mathrm{q}$ is an FCL rule, where $P_1, P_2, \dots P_n$ are obligation propositions and q is a deontic proposition then $P_1 \otimes P_2 \otimes \dots P_n \otimes \mathrm{q}$ is a reparational chain. The *reperational chain* indicates that, if the primary obligation P_1 is violated, its violation can be repaired by the secondary obligation P_2 and if P_2 cannot be satisfied then it can be repaired by the obligation P_3, and so on. The entire rule is evaluated to false, if none of the primary obligation, or any of the reparation deontic propositions (respecting their order) is satisfied.
- Prohibitions can be either represented as $O\neg$ or $\neg P$.

4.4 Formal Specification of Compliance Requirements

This sub-section examines and compares how compliance requirements of the running scenario as described in Table 1 can be formalized in the three languages.

R1		
LTL:	$G(CheckCreditWorthiness.Role(Role1) \rightarrow G($ $(Role1 =' PostProcessingClerk' \vee Role1 = 'Supervisor')))$	(1)
CTL:	$AG(CheckCreditWorthiness.Role(Role1) \rightarrow AG($ $(Role1 =' PostProcessingClerk' \vee Role1 = 'Supervisor')))$	(2)
FCL:	$Role1 \neq' PostProcessingClerk' \wedge Role1 \neq' Supervisior'$ $\vdash O_{Role1} \neg CheckCreditWorthiness$	(3)

This compliance requirement can be represented in the three languages. In FCL, the semantics of the requirement can be captured by prohibiting any other role rather than $PostProcessingClerk$ and $Supervisor$ from performing $CheckCreditWorthiness$ activity. You can notice that that the LTL and CTL representations can be viewed as the contra-positive of the FCL representation ($P \rightarrow Q \equiv \neg P \rightarrow \neg Q$)

R2		
LTL:	$G((CheckCustomerBankPrivilge.Role(Role1) \rightarrow G(\neg(CheckCreditWorthiness.Role(Role1))$	(4)
CTL:	$AG((CheckCustomerBankPrivilge.Role(Role1) \rightarrow AG(\neg(CheckCreditWorthiness.Role(Role1))$	(5)
FCL:	$CheckCustomerBankPrivilge.Role(Role1);$ $CheckCreditWorthiness \vdash O_{Role1}\neg CheckCreditWorthiness$	(6)

This requirement represents the typical segregation of duties constraint and it can be represented in the three languages.

R3		
LTL:	$Can't\ be\ represented$	(7)
CTL:	$Can't\ be\ represented$	(8)
FCL:	$LoanAmount(y) \geq' 1M' \vdash O_{supervisor}CheckCreditWorthiness \otimes O_{System}StoreSuspenseFile(y) \otimes O_{System}notifyManager \wedge P_{PostClerk}CheckCreditWorthiness$	(9)

This requirement can be represented in FCL using the $\otimes$ (CTD) operator. Besides, FCL supports the notion of permission, which is *not* supported in LTL or CTL (e.g. $P_{PostClerk}CheckCreditWorthiness$). Although semantically unequal, the closest operator in temporal logic is the disjunction operator, however, it is commutative.

R4		
LTL:	$\neg SignOfficiallyContract\ W\ (JudgeHighRisk \wedge approved =' Yes')$	(10)
CTL:	$A(\neg SignOfficiallyContract\ W\ (JudgeHighRisk \wedge approved =' Yes'))$	(11)
FCL:	$Can't\ be\ represented$	(12)

This requirement can't be represented in FCL due to its lack of support to temporal operators.

R5		
LTL:	$FG(LoanConditions =' True' \rightarrow ((GF(CheckLoanStatus))\ U\ NotifyCustomer))$	(13)
CTL:	$Can't\ be\ represented$	(14)
FCL:	$Can't\ be\ represented$	(15)

Neither CTL nor FCL can express the *weak fairness* property of *R5* (a constantly enabled event must occur infinitely often) [5], which is expressible in LTL. The same applies to the specification of *strong fairness* properties.

R6		
LTL:	$Can'tbe\ represented$	(16)
CTL:	$AG(CheckCreditWorthiness \rightarrow EF(JudgeHighRisk))$	(17)
FCL:	$Can'tbe\ represented$	(18)
This requirement can only be expressed in CTL due to its support to the existential quantifier.		
R7		
MTL	$G(SendLoanContract \rightarrow F_{\geq 5}(PerformLoanSettlement.Role('CreditBroker')))$	(19)
TCTL	$AG(SendLoanContract \rightarrow AF_{\geq 5}(PerformLoanSettlement.Role('CreditBroker')))$	(20)
FCL	$SendContract: t \vdash O_{CreditBroker}\ PerformLoanSettlement: k \wedge k \geq t{+}5$	(21)
Requirement *R7* is not expressible in LTL or CTL due to their lack of support to real-time requirements. Some extensions to LTL and CTL have been proposed to incorporate real-time dimension. For example, Metrical Temporal Logic (MTL) [11] extends LTL with real-time dimension. In MTL, temporal operators can be annotated with a temporal expression I expressing a specific time interval as shown in the MTL representation of *R7* (e.g. $F_{\geq 5}$). Timed CTL (TCTL) [12] extends CTL with real-time dimension exactly the same way as MTL extends LTL. Temporal dimension is also incorporated to FCL as proposed in [4], such that all propositions can be time-stamped. If we can conclude P at time t, written as $P{:}\,t$, then P is true for all $t' > t$, until an event occurs that terminates the validity of P.		

5 Comparative Analysis between LTL, CTL and FCL

Table 2 summarizes the results of the comparative analysis, which highlights the strengths and limitations of the three languages. The degree of support is denoted by: '+', indicating that the feature is satisfied, '-'; indicating that the feature is not satisfied; and '±', indicating that the support is partial.

Some of these results can be generalized to the whole families of Deontic logic and Temporal Logic. For example, FCL, CTL and LTL possess limitations in terms of *usability*. This result can be generalized to the whole families of Deontic and Temporal Logic. The complexity of logical languages represents one of the main obstacles of utilizing the sophisticated reasoning and analysis tools associated with these languages. FCL, LTL and CTL have different expressive powers. For example, the notion of permission is not expressible in LTL and CTL, while fairness properties are not expressible in FCL and CTL; on the other hand, existential properties are not expressible in LTL and FCL. Deontic and Temporal families of logic are declarative by nature. Furthermore, FCL provides a mechanism for consistency checks by the means of the *superiority relation* of the *defeasible logic* [13], yet this result can't be generalized to the Deontic Logic family (denoted by '?' in Table 2). Temporal Logic family doesn't provide any support for checking consistency among logical formulas. The normalization metric is met by FCL as it provides a technique for cleaning up the specification and to identify and remove redundancies [4].

Table 2. Comparative Analyses of Compliance Request Languages

	LTL/ MTL	CTL/ TCTL	Temporal Logic	FCL	Deontic Logic
1- Formality	+	+	+	+	+
2- Usability	-	-	-	-	-
3- Expressiveness	±	±	?	±	?
4- Declarativiness	+	+	+	+	+
5- Consistency Checks	-	-	-	+	?
6- Non-Monotonicity	±	-	-	+	?
7- Generic	±	±	?	±	?
8- Symmetricity	-	-	?	±	?
9- Normalization	-	-	-	+	?
10- Intelligible feedback	+	±	?	-	-
11- Real-time Support	+	+	?	+	?

Non-monotonic requirements can be expressed in FCL by means of the *superiority relation*. On the other hand, rules in temporal logic are monotonic by nature. In FCL, by exploiting the results in [1], business process models can be visually annotated by compliance requirements using the notion of *control tags*. However, with symmetricity we mean the actual augmentation of business process models with compliance requirements (thus the support for this feature is marked as '**±**' for FCL). Model-checkers are used with temporal logic for automatic compliance verification [6]. As concluded in [10], it is usually possible with LTL to provide the *counterexample tracing* facility that helps experts to resolve a compliance violation, thus providing the user with intelligent feedback. The support to this feature is limited for CTL. In [14], we propose a comprehensive 'root-cause analysis' to reason about design-time compliance violations to provide the user with guidelines as remedies to resolve compliance deviations, which is based on LTL. The support by Deontic logic family to this criterion is limited.

Several extensions to LTL and CTL have been proposed to incorporate real-time dimension (e.g. MTL [11] and TCTL [12]). Real-time dimension is also incorporated to FCL as proposed in [4]. Finally, a basic strength of LTL and temporal logic in general lies in its maturity and availability of sophisticated *verification tools* that have proven to be successful to verify complex systems [10].

Vardi provides an interesting comparison between LTL and CTL in [10]. Although, CTL and LTL correspond to two distinct views of time, and consequently LTL and CTL are expressively incomparable. However, from a practical point of view LTL is considered to be more expressive than CTL. Besides, LTL is considered to be more intuitive than CTL. The un-intuitiveness of CTL significantly reduces the usability of CTL-based formal verification tools. From a verification point of view, CTL is considered to be more difficult than LTL due to the branching nature of CTL. Furthermore, CTL does not provide support for compositional reasoning. The mainadvantage of CTL over LTL is its *computational complexity*. However, Vardi argues that LTL is a more powerful logic and CTL's advantage in terms of computational

complexity is valid under rare circumstances in real life applications. On the other side, the computational complexity of FCL is unknown (compliance verification is based on the *Idealness* notion as proposed in [4]).

6 Related Work

In [15], a comparison is conducted between three types of logics: (i) CL (Contract Language): Deontic logic, (ii) LTL and CTL: temporal logics and (iii) Communicating Sequential Processes (CSP): operational language, with respect to their expressiveness to represent three requirements emerging from a business contract. Although we agree with the conclusion highlighting CL's power to represent the business contract under consideration, we diverge with the argument that states LTL's lack of support to some fairness properties. Although our main focus in this study is on regulatory compliance, the comparative analysis conducted in this paper is more generic and considers an extensive list of comparison criteria in addition to the *expressiveness* metric.

It is also of relevance here to summarize various key studies that utilize temporal and deontic logic for design-time compliance verification. Authors in [16] propose a static-compliance checking framework that includes various model transformations. Compliance requirements are based on LTL formulas. Next, NuSMV2 model checker is used to check the compliance. The study in [17] utilizes π-Logic to formally represent compliance requirements; while BP models are abstractly modeled using BP-Calculus. If business and compliance specifications are compliant, an equivalent BPEL program can be automatically generated from the abstract BP-calculus representations. The study in [18] utilizes past LTL (PLTL), where properties about the past can be represented. The study in [19] utilizes patterns to overcome the complexity of temporal logic focusing on runtime monitoring. The study in [20] utilizes Dwyer's patterns for the verification of service compositions. In [21], real-time temporal object logic is proposed for the formal specification of compliance requirements based on a pre-defined domain ontology. In [22], we use LTL and proposes a framework for augmenting business processes with reusable fragments to ensure process compliance to the relevant requirements by design.

We have to point out that there is a third class of languages that can be used for the formal specification of compliance requirements grounded on XML, e.g. the XML Service Request Language (XSRL) [23] and the PROPOLS language [20]. Since XML-based approaches are founded on temporal logic, then they are subsumed by LTL and CTL, subsequently, they are not considered in our analytical study.

Key studies that have utilized Deontic logic can be summarized as follows: the work in [4] has provided the foundations of the FCL (Formal Contract Language) language, focusing on business partner contracts. In addition, in [13], an automatic transformation of business contracts represented in FCL to RuleML is proposed for runtime monitoring. In [1], FCL is used to express other types of compliance requirements emerging from legislation and regulatory bodies. In [24], the PENELOPE (Process Entailment from the Elicitation of Obligations and PErmissions) language was proposed. PENELOPE is a language to express temporal deontic assignments considering only obligations and permissions.

7 Conclusion and Outlook

An important question that might arise in the field of compliance management is: "How compliance requirements can be formally specified to enable the application of automatic analysis and reasoning technique for their verification?" Temporal and deontic families of logic have been successfully utilized in the literature as the formal foundation of compliance requirements. In this paper, we report a comparative analysis between LTL, CTL and FCL. The comparison surfaces the strengths and limitations of each language with respect to a set of identified features. Some of these conclusions can be generalized to the whole family of temporal or deontic logic. The decision on the use of a particular formal language is context-dependent that should be based on the nature, complexity and source of compliance requirements. However, based on the overall findings of the comparative analysis as well as the relevant literature and the current practice, we argue that temporal logic has advantages over other formalisms under consideration when formal specification of regulatory compliance requirements is concerned. An important strength in temporal logic is its maturity and its sophisticated tool support.

It should also be noted that the identified comparison criteria are not equally important. For example, the support of temporal logic to the *intelligible feedback* and *sophisticated tool support metrics* is significant. We also agree with Vardi's argument in [10] that LTL is a more powerful logic. CTL* is the logic that combines the expressive power of LTL and CTL, however, its computational complexity is *2PTime-Complete*.

An interesting ongoing research direction is to resolve the main problems of LTL that have surfaced from the comparative analysis. In particular, developing a graphical language tool based on recurring property patterns [25] relevant to the compliance context would address the usability metric. Besides, providing efficient solutions to support the specification of non-monotonic rules in LTL, as well as normalization and consistency checking are other areas for future research. Finally, analyzing and investigating these languages on the basis of the support they provide not only for design-time verification but also for runtime monitoring -hence, integrating these two phases and providing a lifetime compliance management support - is an important ongoing research direction.

Acknowledgments. We express our sincere thanks to PricewaterhouseCoopers (Netherlands) and Thales Services SAS (France) for their effort in providing and participating in the case studies and scenarios, and their valuable contributions.

References

1. Sadiq, S., Governatori, G., Naimiri, K.: Modeling Control Objectives for Business Process Compliance. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 149–164. Springer, Heidelberg (2007)
2. Papazoglou, M., Traverso, P., Dustdar, S., Leymann, F.: Service-Oriented Computing: State of the Art and Research Challenges. Computer 40, 38–45 (2007)
3. Thao, L., Goser, K., Rinderle-Ma, S., Dadam, P.: Compliance of Semantic Constraints-A Requirements Analysis for Process Management Systems. In: 1st GRCIS 2008 Workshop, France, pp. 41–45 (2008)
4. Governatori, G., Milosevic, Z., Sadiq, S.: Compliance Checking Between Business Processes and Business Contracts. In: 10th EDOC 2006, Hong Kong, pp. 221–232 (2006)

5. Pnueli, A.: The Temporal Logic of Programs. In: 18th IEEE Symposium on Foundations of Computer Science, Providence, pp. 46–57 (1977)
6. Clarke, E., Grumberg, J., Peled, D.: Model Checking. MIT Press, Cambridge (2000)
7. COMPAS official web site – Project description, http://www.compas-ict.eu/project.php
8. COMPAS Project, D 2.1, State-of-the-Art in the Field of Compliance Languages (2008)
9. COMPAS Project, D 6.1, Use Case, Metrics and Case Study Definition (2008)
10. Vardi, M.: Branching vs. Linear time: Final showdown. In: Margaria, T., Yi, W. (eds.) TACAS 2001. LNCS, vol. 2031, pp. 1–22. Springer, Heidelberg (2001)
11. Alur, R., Henzinger, T.: Real-time Logics: Complexity and Expressiveness. Information and Computation 104, 35–77 (1993)
12. Alur, R.: Techniques for Automatic Verification of Real-time Systems. vol. Ph.D. thesis. Stanford University (1991)
13. Governatori, G.: Representing Business Contracts in RuleML. International Journal of Cooperative Information Systems (2005)
14. Elgammal, A., Turetken, O., van den Heuvel, W., Papazoglou, M.: Root-cause analysis of design-time compliance violations on the basis of property patterns. In: Maglio, P.P., Weske, M., Yang, J., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6470, pp. 17–31. Springer, Heidelberg (2010)
15. Fenech, S., Okika, J., Pace, G., Ravn, A., Schneider, G.: On the Specification of Full Contracts. In: 6th FESCA 2009 workshop, UK (2009)
16. Liu, Y., Muller, S., Xu, K.: A Static Compliance-Checking Framework for Business Process Models. IBM Systems Journal 46 (2007)
17. Abouzaid, F., Mullins, J.: A Calculus for Generation, Verification, and Refinement of BPEL Specifications. In: 3rd WWV 2007 Workshop, Italy, pp. 43–68 (2007)
18. 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)
19. Namiri, K., Stojanovic, N.: Pattern-based Design and Validation of Business Process Compliance. In: Chung, S. (ed.) OTM 2007, Part I. LNCS, vol. 4803, pp. 59–76. Springer, Heidelberg (2007)
20. Yu, J., Manh, T., Han, J., Jin, Y.: Pattern-Based Property Specification and Verification for Service Composition. In: Aberer, K., Peng, Z., Rundensteiner, E.A., Zhang, Y., Li, X. (eds.) WISE 2006. LNCS, vol. 4255, pp. 156–168. Springer, Heidelberg (2006)
21. Giblin, C., Liu, A., Muller, S., Piftzmann, B., Zhou, X.: Regulations Expressed As Logical Models. In: 18th JURIX 2005, Belgium, pp. 37–48 (2005)
22. Schumm, D., Turetken, O., Kokash, N., Elgammal, A., Leymann, F., van den Heuvel, W.: Business process compliance through reusable units of compliant processes. In: Daniel, F., Facca, F.M. (eds.) ICWE 2010. LNCS, vol. 6385, pp. 325–337. Springer, Heidelberg (2010)
23. Lazovik, A., Aiello, M., Papazoglou, M.: Planning and Monitoring the Execution of Web Services Requests. International Journal on Digital Libraries 6, 235–246 (2006)
24. Goedertier, S., Vanthienen, J.: Designing Compliant Business Processes with Obligations and Permissions. In: Eder, J., Dustdar, S. (eds.) BPM Workshops 2006. LNCS, vol. 4103, pp. 5–14. Springer, Heidelberg (2006)
25. Dwyer, M., Avrunin, G., Corbett, J.: Property Specification Patterns for Finite-State Verification. In: 2nd International Workshop on Formal Methods on Software Practice, USA, pp. 7–15 (1998)

Performance and Cost Assessment of Cloud Services

Paul Brebner[1] and Anna Liu[2]

[1] (NICTA/ANU)
[2] (NICTA/UNSW)
{Paul.Brebner,Anna.Liu}@nicta.com.au

Abstract. Architecting applications for the Cloud is challenging due to significant differences between traditional hosting and Cloud infrastructure setup, unknown and unproven Cloud performance and scalability characteristics, as well as variable quota limitations. Building workable cloud applications therefore requires in-depth insight into the architectural and performance characteristics of each cloud offering, and the ability to reason about tradeoffs and alternatives of application designs and deployments. NICTA has developed a Service Oriented Performance Modeling technology for modeling the performance and scalability of Service Oriented applications architected for a variety of platforms. Using a suite of cloud testing applications we conducted in-depth empirical evaluations of a variety of real cloud infrastructures, including Google App Engine, Amazon EC2, and Microsoft Azure. The insights from these experimental evaluations, and other public/published data, were combined with the modeling technology to predict the resource requirements in terms of cost, application performance, and limitations of a realistic application for different deployment scenarios.

Keywords: Cloud performance, scalability, cost, limits, quotas, service-oriented performance modeling (SOPM), Amazon EC2, Google AppEngine, Microsoft Azure.

1 Introduction

Since 2007 NICTA has developed Service-Oriented Performance Modeling (SOPM), a technology for modeling the performance and scalability of Service Oriented Architecture (SOA) applications. SOPM has been trialed and proven in both laboratory settings and with government and commercial collaborators [1-4]. Recently we have examined a sub-class of SOA applications deployed on Virtualized or Cloud platforms. Some initial results focusing on power costs and carbon emissions were published in [5]. In this earlier work we assumed that the performance of in-house and Cloud deployments of an application were identical. In this paper we revisit this assumption based on a the results of running a suite of empirical tests on Amazon, Google and Microsoft cloud platforms, combined with other published and public insights into cloud performance and scalability.

E.M. Maximilien et al. (Eds): ICSOC 2010 Workshops, LNCS 6568, pp. 39–50, 2011.

2 Service-Oriented Performance Modelling Clouds

SOPM is a method with tool support for modeling SOA application performance and scalability. It supports the direct modeling of software, usage, and resourcing in terms of services (externally and internally visible, 3rd party; compositions and simple services), workloads (external systems which consume the services), and the deployment of services on physical resources (servers, networks). Models are parameterized with workload, performance, and server data, and a model-driven approach automatically generates a run-time version which is solved by a discrete event simulator to compute workload, service, and server metrics to assist with answering performance questions and identify areas of performance risk. We have found that modeling SOA performance is valuable and complements testing as modeling can often be done cheaper, before a complete system is built, when testing is impractical, and can be used to rapidly investigate architectural alternatives.

The results of this paper are based on modeling a realistic SOA application based on a real system that was previously modeled and validated (a whole-of-government common e-Business service used across many different agencies). We modeled two different workloads over a typical 24 hour period. For a number of different hosting scenarios our goal was to model and compare resource requirements for different in-house and cloud platforms (Section 3), performance differences between platforms, and variability in performance within a platform (Section 4). Resource requirements were used to estimate power and billing costs (Section 3), and check if any limits/quotas are exceeded (Section 5).

The alternative hosting scenarios for comparison are: In-house hosting on bare hardware, in-house hosting with Virtualization, Amazon EC2 (a variety of instance types), Google AppEngine, and Microsoft Azure. There are many similarities (architectural, technological, billing, etc) but also significant differences between platforms. We have developed performance models for each cloud platform using data from a variety of sources. Insights and the results from a suite of test applications we ran ourselves were combined and confirmed with data from the public domain including other papers, public blogs and websites, cloud platform support blogs, benchmarks, and real-time cloud monitoring sites.

3 Resources and Cost

Scenario 1: In-house hosting, bare hardware. The first scenario is the default hosting environment from which the performance measurements were obtained to parameterize and validate the initial model. The services from each of the four deployment zones are deployed to four dedicated physical servers (clustered if necessary). The servers are identical multi-core Intel 2.33GHz CPU rack servers with sufficient memory and LAN bandwidth to ensure that there are no bottlenecks, and "perfect" hardware-based load balancing across clustered servers in the same zone. A fundamental assumption is that the application has been architected and the infrastructure configured to be linearly scalable by increasing CPUs and servers (scale-up and scale-out). We also assume that the performance measurements used to parameterize the model have been obtained from an unloaded system which does not have variable speed CPUs (or the results have been scaled for the difference in CPU speed between

light and heavy loads). Running the simulation model with the expected load predicted the maximum number of CPUs required (without any resource saturation or degradation in service response times) for each zone as follows: Web Server 10 CPUs, Client Server 4 CPUs, Security Server 3 CPUs, and Application Server 26 CPUs, a total of 43 CPUs.

Scenario 2: In-house hosting, virtualization. The second scenario is in-house hosting but on virtualized hardware. The performance model is identical to scenario 1, except that each application zone is deployed to a virtual machine, with the virtual machines sharing a pool of CPUs. We assume that the servers are identical to scenario 1, but that the use of virtualization incurs a performance overhead. Obviously the value of the overhead depends on a combination of factors including CPU type, virtual machine type, application, and load. For this experiment we assumed a CPU overhead of 70% for both workloads on Intel CPUs due to IO virtualization [6]. The model predicted a maximum of 70 CPUs in the pool which is slightly less than 1.7 times the maximum CPUs from scenario 1, as CPU pooling is more efficient overall and ensures maximum utilization. The introduction of virtualization in theory enables easier sharing of the CPU pool with other applications during off-peak periods, although care needs to be taken that there are sufficient CPUs for this application during peak times.

Scenario 3: Cloud Hosting, Amazon EC2. This scenario explores hosting the complete application on the Amazon Elastic Compute Cloud (EC2) infrastructure [9]. For simplification we make the following assumptions about the EC2 hosting: All services are deployed to each instance (in practice different services may need to be deployed to different instances. We look at the performance implications of distributed deployment, but not the resource implication, in Section 4); all instances are of the same type (this is not a requirement of the infrastructure, but simplifies modeling and may be a reasonable simplification for managing clouds in practice); Elastic Load Balancing (ELB) is used, but not modeled (as we cannot determine how many instances are required as load balancing instances scale automatically [7]); auto-scaling of EC2 instances is enabled and there is no delay in spinning up new instances [8] (in practice it takes time to start new instances).

We assume that Amazon Auto Scaling is used to start and terminate instances sufficiently in advance to be available when the load requires them (but only just in time, and that they are terminated immediately after the load drops). Elastic Load Balancing and Autoscaling are relatively complex, and need to be setup (and tuned) for optimal scalability. A significant assumption is that Elastic Load Balancing instances dynamically scale fast enough to cope with load spikes (otherwise clients may experience time outs). Other considerations include ensuring that the number of instances in each availability zone is equivalent, clients are from a variety of IP addresses rather than just one (clients from the same IP address tend to connect to the same instance), and the use of persistent HTTP connections to improve performance. In this section we focus on modeling resource usage, rather than general performance/scalability issues (Section 4).

Amazon EC2 instances come in different types [9]. Each type offers a total number of standardized EC2 Compute Units per instance, consisting of a number of cores per instance, number of compute units per core, and at different prices. A core is a virtual core, but virtual cores have exclusive use of a physical core (except standard small instances where two virtual cores share a physical core). Our model must therefore take

into account the difference in performance between EC2 instance types. However, we also observe that not all EC2 instances of the same type are equal, as there are performance differences between instances of the same type. EC2 instances of the same type vary in performance depending on region. We augment the initial SOA performance model with a scaling factor to capture the difference in performance between bare hardware and the target EC2 instance type. We rely on observations that there is a difference in speed between Intel and AMD CPUs, that CPUs have a different speed, and that EC2 instances of the same type are made up of a proportion of both Intel and AMD CPUs. [11-15]. Including the virtualization overhead scale factor from scenario 2 (EC2 uses Xen virtualization) we introduced scale factors for each instance type into the model and obtained estimates of resource requirements in terms of maximum number of EC2 instances (Figure 1), and total EC2 instance hours per day (not shown).

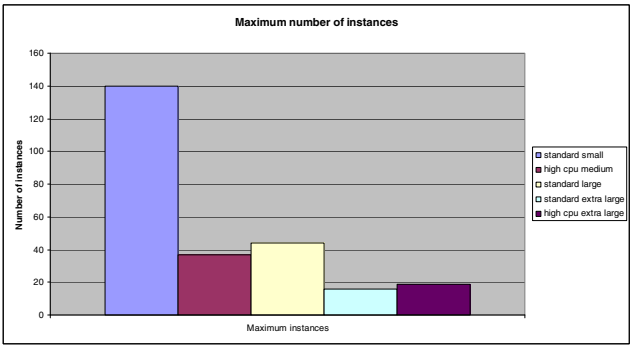


Fig. 1. Maximum EC2 Instances

Amazon offers three pricing options [16]: On-demand, reserved, and spot prices. Billing is always per instance hour (rounded up). We computed and compared two different options. Option 1 is all on-demand pricing. Option 2 is a mixture of reserved instances (for base load) and spot instances (for peak loads). As spot instances can be terminated without warning [19] the application needs to be re-designed to be fault tolerant. On-demand prices were based on an average of windows instances for

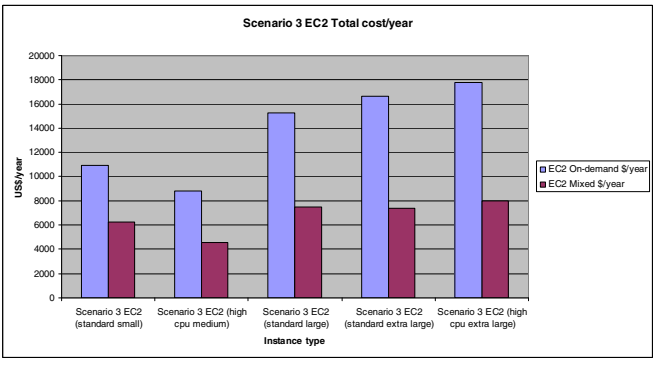


Fig. 2. EC2 cost per year

all regions, reserve prices were California (Linux only available), and spot prices were based on an average price over several months [17]. Including network and CloudWatch [18] (necessary for Auto Scaling) prices (a total of $1,300/year) the total estimated costs per year are graphed in Figure 2. There are significant price differences between instance types, and between pricing options. The most expensive is on-demand EC2 high cpu extra large instances (US$18,000) and the cheapest is mixed price high cpu medium instances (US$4,500).

Scenario 3: Cloud Hosting, Google AppEngine. The Google AppEngine [20] uses a different approach to EC2 for resourcing as the infrastructure automatically scales. It doesn't use virtualisation (although Java applications run in a Java Virtual Machine giving application isolation but not resource isolation). Consequently there are limitations to what can be done by an application to ensure isolation (e.g. no threads, only a subset of the Java APIs are permitted), and there is less visibility into the infrastructure for modelling. Billing is in terms of a reference CPU (1.2GHz Intel), but as we could not determine what the physical hardware speed was we assumed it was the same speed and type of CPU as the in-house model. We assumed that automatic scale-up is instantaneous (which is unlikely, and there is no mechanism available to ensure resources are available in advance of load spikes [23]). CPU billing is finer grained as it is measured cycles and reported in seconds. There are also more complex limits/quotas, and free and billable thresholds [21, 22].

The model estimated 44.8 physical (not reference) CPU hours/day, and the total price per year including data costs ($3,000) of US$4,723/year which is comparable to the cheapest EC2 option. The main benefit with Google AppEngine pricing is that it is very fine grained, so you really do only pay for what you consume (per CPU cycle), and there's no overhead (cost or effort) to scale and manage it.

Scenario 3: Cloud Hosting, Microsoft Azure. The Microsoft Azure platform provides a similar resource model to Amazon EC2. It uses virtualization, has different instance sizes (with 1, 2, 4, or 8 cores), and is hosted on 1.9GHz AMD CPU's. Billing is per instance hour (rounded up) for 1.6GHz reference CPUs (for simplicity we assume billing is really done for 1.9GHz CPUs as this was the speed of CPUs used). Azure offers different instance roles: Web (external facing clients), worker (back-end), and AppFabric (orchestration, security, service bus etc). For resource estimation we assume that only Web instances and workers are used, but for pricing we include

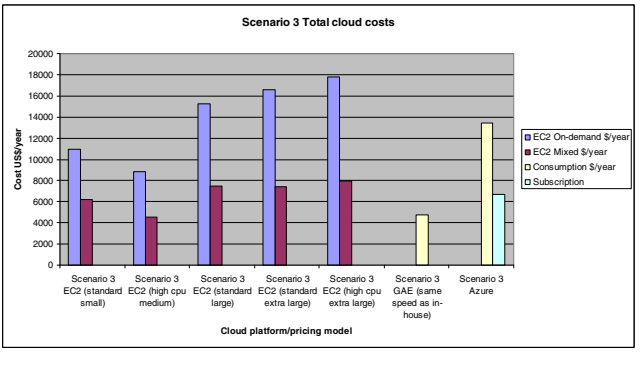


Fig. 3. Cloud scenarios cost comparison

AppFabric access control (transaction) and service bus (connection) costs, and data costs ($4,500), giving a total price of US$13,417/year. Data cost was based on Asian area costs, but ignored difference in peak/off-peak times. Microsoft has a number of promotional offers such as a 6 month subscription which may give up to a 50% saving compared with consumption rates, reducing the yearly cost to $6700, on par with some of the EC2 mixed pricing options (Figure 3).

4 Performance

For performance modeling we wanted to capture some aspects of the difference and variability in performance between and within cloud platforms and in-house hosting. Rather than modeling these directly as a consequence of resource limitations, we only attempted to model them in terms of increases in times based on assumptions about times and distributions. We will examine these in terms of Server, LAN, and WAN times.

Server performance variability. The majority of cloud platforms use virtualization to ensure that applications are isolated from each other, and to guarantee a minimum dedicated allocation of resources to each virtual core. EC2 and Azure use virtualization, and we assume that each virtual core is given a dedicated physical core (except EC2 small instances which has half a physical core). However, this only guarantees CPU. Virtual machines on the same server compete for other resources including memory and IO bandwidth. We assume that for EC2 and Azure the virtualization is "good enough" to ensure reasonably consistent performance. However, for Google AppEngine, which does not use virtualization but some other resource scheduler, we introduced an intermittent (1% chance) performance overhead of up to 5 times to capture observations about longer time due to competition for resources and as an artifact of the CPU scheduler [25]. This may be an underestimate of performance variability as some results suggest up to 50 times longer times in 20% of cases [24].

LAN latency and bandwidth. For resource modeling (Section 3) we assumed that the entire application is deployed in each virtual machine instance. However, in practice this is unlikely to be feasible in all situations, and a distributed application will be necessary [32]. This may impact on resource usage (but minimally, as the peak usage is the dominant factor), but will definitely impact performance due to inter-component communication latencies. Studies have reported good (but not as high as theoretically available) LAN bandwidth between EC2 machines [27], but the latency is significant and increases when multiple demanding applications are sharing the same LAN segment or server, when instances aren't in the same geographical location, and when there is high packet loss [26, 28, 29]. The typical inter-instance latency (in the same zone) is between 1ms and 1s (average 50ms). We introduced this latency distribution to the model for every interaction between services on different instances, and a delay for the amount of data transferred between instances (assuming 500Mbps bandwidth). This is only an attempt to model the average delays, as in the worst case the LAN may be heavily congested with latencies increasing dramatically for extended periods of times. Users have reported that they have to kill instances and start new instances in an attempt to find servers that work better [30]. The latency also probably depends on the inter-instance protocol used. For example, AWS Queue

service latency is reported at 1.5s [31], making this a poor choice of inter-instance protocol for systems where service times are sub-second.

Figure 4 shows the minimum and maximum response times predicted (across all three external services) for selected scenarios. The in-house hosting has the best times of between 613ms and 16.9s, with Cloud hosting options showing an increase in minimum response times (1.9s seconds for EC2 small instances) and maximum response times (Google AppEngine 68.5 seconds). Assuming the system is required to support a SLA with a maximum response time for any service of 30s, none of the cloud options are compliant.

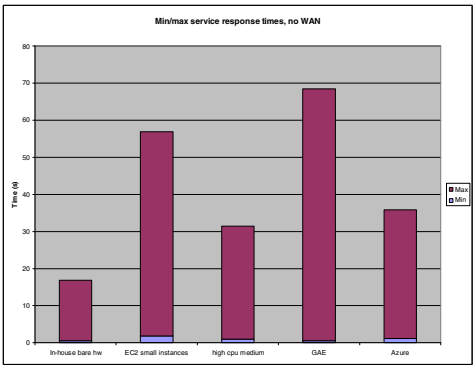


Fig. 4. Minimum and maximum service response times

WAN latency and bandwidth. The above performance results ignored WAN latency and bandwidth aspects. We added latency and transfer speeds to the model, initially assuming 100Mbps WAN bandwidth for data transfers for both continental (in-house), and inter-continental data transfers (for cloud scenarios), and identical transfer rates in both directions (Figure 5).

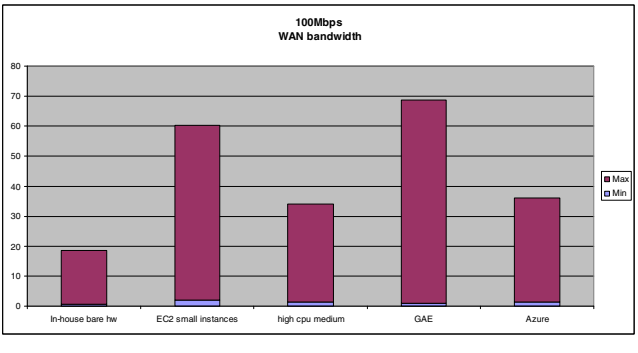


Fig. 5. Response times with 100Mbps WAN

However, based on our own measured results and 3rd party monitoring information we quickly concluded that because of large inter-continental latencies, high theoretical bandwidths, non-zero packet loss, and asymmetries in upload and download

speeds, a more achievable upload speed is around 0.5Mbps with download speed only marginally better at 1Mbps. The predicted maximum response time for in-house hosting with 100Mbps bandwidth WAN (continental) is 18s, but a whopping 271s (almost 5 minutes) for USA hosted EC2 (inter-continental). For the slowest service (external service "3" with 10Mb file) the WAN transfer time makes up the bulk of the response time (88%).

This is a stark demonstration of the so-called "Skinny Straw" problem with Clouds [33] – the fact that high latency, high bandwidth networks don't achieve anywhere near their maximum theoretical speed with standard TCP protocols and configurations.

5 Platform Limits and Quotas

In this section we explore which scenarios would actually work correctly given published platform limits and quotas. EC2 has a limit of 20 concurrent on-demand or reserved instances, or 100 spot instances. For the on-demand price simulations, only the standard extra large and high cpu extra large instances are under the 20 on-demand instance limit. Using mixed (reserved and spot-instances) pricing model, all instance types except small instances are under the 100 spot instance limit (Figure 1).

Because Google AppEngine doesn't use virtualization to isolate applications/users there is a risk that some applications will act as resource hogs and attempt to use more than their "fair" share of resources. The approach AppEngine takes to limit this behaviour is to impose multiple limits/quotas on each user. These are relatively complex to understand, and include total, daily, minute and instantaneous limits. Users can request an increase in limits/quotas, but to do so they need to be able to estimate which limits will be exceeded and the new limits to request as follows:

- AppEngine permits a maximum of 30 concurrent requests. The model predicts a maximum of 59 concurrent users (for all services), exceeding the limit.
- The maximum CPU consumption rate is 72 CPU minutes/minute (in terms of reference CPU times), and the model predicts 40 CPUs (physical), so multiplying by 2 gives 80 reference CPU minutes/minute, exceeding the limit.

For Azure subscription accounts there is a maximum of 20 hosted service projects, 5 roles per hosted service, and 20 VM CPU Cores [35]. The limit of 20 VM CPU Cores is exceeded as 72 Cores are predicted by the model for the peak load.

6 Observations and Conclusions

Our modeling approach, and many of the cloud platforms, give good visibility into the operational costs of an application, and can capture costs for various resource types and load such as the cost of CPU (total, and due to different workloads and base and peak loads), network/data, security operations, transactions, orchestration, connections, etc.

Different cloud charging and billing models allows choice of hosting options between and even within cloud platforms. However, this adds both flexibility and complexity, which modeling may assist with.

At face value a consumption-based charging model can reduce costs. However, it may also focus too much attention on cost as an architectural driver. Optimizing applications for very specific costing models may result in vendor lock in and lack of flexibility and maintainability.

Is cloud hosting cost effective? Our predictions are that cloud costs are comparable to the cost of power alone for in-house hosting. However, our pricing was not exhaustive, and there are likely to be other hidden costs. The final decision about cost/risk benefits must include the cost of migrating applications to the cloud, added maintenance and management complexity due to remote hosting, increased chance of outages due to failures in cloud and network infrastructures (and therefore loss of customer confidence and possibly financial penalties for SLA violations), and importance and economic return of the services offered.

Does performance matter? From experiments and modeling it is obvious that cloud performance is likely to be worse and substantially more variable than in-house hosting. But is it "good enough"? This obviously depends on the type of application, if there are hard response time requirements for services (e.g. due to constraints in the physical world or software of the consumers), or if SLAs must be satisfied. For user facing applications there are maximum response times above when users are impacted and user behavior and satisfaction changes, resulting in abandonment or complaints. Depending on the importance of the task, prior expectations, and availability of alternatives, these times can range from seconds to tens of seconds. Some of these issues can be reduced by changing the way users interact with the services (e.g. allowing asynchronous interactions, providing constant feedback about time to completion, allowing concurrent interactions, etc). For machine-to-machine interactions there are issues such as client-side timeout settings and retry behavior (e.g. which may result in increased load on the services and duplicate transactions).

In most situations it is valuable to offer explicit SLAs per service, and the modeling approach assists with this. Another approach is to offer tailored SLAs for different classes of users of a set of services. This could be enabled by the use of virtualization in most cloud platforms. A set of services could be deployed on a virtual machine and configured and resourced in such a way that a SLA can be guaranteed for a sub-class of users. Fractional and elastic resourcing would assist with ensuring the SLA would be met for a variety of loads.

It may be non-trivial to architect distributed applications for performance and scalability, as the interactions between different components may be complex (and inter-instance latency combined with the number of interactions can be a significant overhead), and resource requirements for each component may differ substantially as loads change. There may be complications with load-balancing across multiple instances of the same component type, and applications will need to be architected to be reliable with lower availability than normal. There may also be components (e.g. databases, 3rd party services, legacy services) that just do not scale well, or have fixed maximum throughputs. Hybrid applications (hosted on both in-house and external clouds) present challenges, particularly due to the cost and low speed of data transfer between distributed components in and out of the cloud (e.g. data will need to be moved to the cloud once and then used in the cloud as much as possible before moving back to the enterprise), and deciding if (and which) components can sensibly be hosted on the cloud (e.g. due to dependencies and inter-connectively to other

systems, security and privacy, difference in performance between enterprise and cloud hosting, etc).

SLAs offered by cloud providers are weak, limited in scope, and don't guarantee availability of all resource types. For EC2 and Azure a dedicated number of physical CPUs (or part therefore for EC2 small instances) is all that is guaranteed. Better SLAs would be valuable for network and IO resources to deal with problems of resource contention in cloud platform infrastructure and applications sharing the same servers. For Google AppEngine, which does not use virtualization, there is no guarantee even of CPU (particularly if the application is CPU intensive as it's priority may be reduced), response times may blow out (due to competing applications, if an application is only intermittently used and is put to sleep, and the first time a new instance of an application is used (e.g. due to JVM startup, lazy loading of application code and data). On the other hand, multiple (complex) limits are imposed in order to attempt to reduce resource contention and limit the impact of greedy applications. Unfortunately, for enterprise applications, the risk of un-predictable performance and exceeding limits (many of which are difficult to relate to physical values, such as CPU time, resulting in application exceptions and periods of unavailability until resources are replenished or can be raised upon request) are likely to be too restrictive. The Google AppEngine programming model may also be too restrictive for enterprise applications as only a subset of Java APIs and enterprise Java Edition technologies are supported [36]. Enterprise Java Beans (EJBs) are not supported due to lack of SQL support, although JDO/JPA (which maps data objects to datastore entities [37]) are supported, and may provide an alternative and scalable persistence model for the cloud.

The ability of cloud platforms to ramp-up with increased load was not tested or modeled. If a cloud platform supports the ability to control the number of instances in use and start/stop instances on demand, then this adds complexity for monitoring and managing to ensure that instances are available when needed. If elasticity is automatic (E.g. Google AppEngine, or Amazon Elastic Load Balancing) then it is important to know what the limitations are through SLAs with the cloud provider and performance evaluations and modeling, so that SLAs with clients can be managed and approaches to limit load rate increases considered.

Modeling is a potentially powerful tool to understand and compare performance, scalability, cost benefits, and risks of various cloud platforms, and hosting and deployment options. If parameterized with performance data obtained from real cloud platforms the models will provide more accurate insights into performance, costs, benefits and risks of critical architectural options than existing Cost of Ownership calculators (e.g. provided by Azure). Further experimentation and research is required to determine if cloud platforms are as scalable as assumed, and if elastic infrastructures can scale quickly enough to cope with load spikes. Unpredictable periods of reduced availability also need to be reckoned with. We plan to enhance our modeling environment to enable direct modeling and simulation of more dynamic properties of clouds platforms.

Acknowledgements

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

1. Brebner, P., O'Brien, L., Gray, J.: Performance Modeling for e-Government Service Oriented Architectures. In: Experience Report Proceedings, ASWEC 2008, pp. 130–138 (2008)
2. Brebner, P.: Performance modeling for service oriented architectures. In: ICSE Companion 2008, pp. 953–954 (2008), doi: http://doi.acm.org/10.1145/1370175.1370204
3. Brebner, P., O'Brien, L., Gray, J.: Performance Modeling Evolving Enterprise Service Oriented Architectures. In: WISA/ECSA 2009, pp. 71–80 (2009), doi: 10.1109/WICSA.2009.5290793
4. Brebner, P.: Service-Oriented Performance Modeling the MULE Enterprise Service Bus Loan Broker Application. In: SEAA 2009, pp. 404–411 (2009), doi:10.1109/SEAA.2009.57
5. Brebner, P., O'Brien, L., Gray, J.: Performance Modeling Power Consumption and Carbon Emissions for Server Virtualization of Service Oriented Architectures. In: Proceedings of the IEEE EDOC 2009 Workshops, Middleware for Web Services Workshop 2009, pp. 92–99 (2009), doi:10.1109/EDOCW.2009.5332010
6. Wood, T., Cherkasova, L., Ozonat, K., Shenoy, P.: Predicting Application Resource Requirements in Virtual Environments. HP Laboratories, Technical Report HPL-2008-122 (2008), http://www.hpl.hp.com/techreports/2008/HPL-2008-122.html
7. Amazon Elastic Load Balancing, http://docs.amazonwebservices.com/ElasticLoadBalancing/latest/DeveloperGuide/
8. Amazon Auto Scaling, http://aws.amazon.com/autoscaling/
9. Amazon EC2, http://aws.amazon.com/ec2/
10. EC2 Instance Types, http://aws.amazon.com/ec2/instance-types/
11. Amazon EC2 benchmark, http://www.mnxsolutions.com/blog/linux/amazon-ec2-benchmark-pystone.html
12. Testing Cloud Computing Performance with PRTG: Performance Comparison of Amazon EC2 Instance Types, http://www.paessler.com/blog/2009/04/03/prtg-7/testing-cloud-computing-performance-with-prtg-performance-comparison-of-amazon-ec2-instance-types/
13. Lamp performance on the elastic compute cloud: benchmarking drupal on amazon ec2, http://www.johnandcailin.com/blog/john/lamp-performance-elastic-compute-cloud:-benchmarking-drupal-amazon-ec2
14. MySQL on EC2 Part 1: are all instances equal?, http://www.infibase.com/blog/2009/07/mysql-on-amazon-ec2-part-1/
15. Walker, E.: Benchmarking Amazon EC2 for high-performance scientific computing, http://www.usenix.org/publications/login/2008-10/openpdfs/walker.pdf
16. EC2 Pricing, http://aws.amazon.com/ec2/pricing/
17. EC2 spot price history, http://cloudexchange.org/
18. EC2 Cloudwatch, http://aws.amazon.com/cloudwatch/
19. EC2 Spot instances, http://aws.amazon.com/ec2/spot-instances/
20. Google AppEngine, http://code.google.com/appengine/
21. Google AppEngine billing, http://code.google.com/appengine/docs/billing.html
22. Google AppEngine quotas and limits, http://code.google.com/appengine/docs/quotas.html

23. Google AppEngine - a second look, http://highscalability.com/google-appengine-second-look
24. Twitmark ported OFF Google AppEngine, http://mrblog.org/2009/10/16/twitmart-ported-off-of-google-app-engine/
25. CPU (Fibonacci) Latency, http://code.google.com/status/appengine/detail/serving-java/2010/01/30#ae-trust-detail-cpu-fibonacci-java-latency
26. Barker, S., Shenoy, P.: Empirical Evaluation of Latency-sensitive Applications Performance in the Cloud, http://none.cs.umass.edu/papers/pdf/mmsys10-latency.pdf
27. Alexandru-Dorin, G.: Network performance in virtual infrastructures – A closer look at Amazon EC2, http://staff.science.uva.nl/~delaat/sne-2009-2010/p29/presentation.pdf
28. Measuring EC2 system performance, http://tech.mangot.com/roller/dave/entry/ec2_variability_the_numbers_revealed
29. EC2 Benchmarks, http://www.wafflegrid.com/wiki/index.php?title=Ec2_Benchmarks
30. http://alan.blog-city.com/has_amazon_ec2_become_over_subscribed.htm
31. http://cloudstatus.com/
32. Are Clouds ready for large distributed applications?, http://www.cs.cornell.edu/projects/ladis2009/papers/sripanidkulchai-ladis2009.pdf
33. Bolden, B.: The Skinny Straw: Cloud Computing's Bottleneck and How to Address it, http://www.cio.com/article/499137/The_Skinny_Straw_Cloud_Computing_s_Bottleneck_and_How_to_Address_It
34. Aspera on-demand, http://www.asperasoft.com/en/products/on_demand_17/aspera_on_demand_for_AWS_17
35. Azure instance limits, http://blog.toddysm.com/2010/01/windows-azure-role-instance-limits-explained.html
36. Google AppEngine support for Java, http://groups.google.com/group/google-appengine-java/web/will-it-play-in-app-engine
37. Google AppEngine datastore, http://code.google.com/appengine/docs/java/datastore/overview.html

Towards Assessing Performance in Service Computing

Claudia-Melania Chituc[1,2]

[1] Faculty of Engineering, University of Porto, Department of Informatics Engineering
[2] LIACC – Artificial Intelligence and Computer Science Laboratory, University of Porto
FEUP-DEI, Rua Dr. Roberto Frias, 4200-365 Porto, Portugal
cmchituc@fe.up.pt

Abstract. Enterprises increasingly choose to focus on core competences and often outsource different services in order to achieve set goals. Although extensive research and development work is pursued on service computing, the main focus is on technical aspects. Business and economic issues in the context of service computing receive little attention. The scope of this article is to present preliminary results of an on-going research project aiming at advancing research in the area of economic performance assessment in service computing by developing a conceptual framework and metrics useful for economic performance analysis. Issues addressed in this article pertain to synthesize and discuss economic theories and approaches relevant for modeling and assessing the economic performance in service computing (e.g., game theory, graph theory, transaction cost economics, decision theory), emphasizing their strengths and weaknesses. Research challenges are then briefly discussed.

Keywords: Service computing; performance assessment.

1 Introduction

Services represent a major part of the IT industry. Nowadays enterprises increasingly prefer to focus on their core areas and often outsource services in order to attain their goals. Services computing support services' modeling, creation and management.

Extensive research is currently being pursued on service computing (and cloud computing). However, most research and development work focuses on technical aspects related to services. So far, relatively scarce research work has been pursued on services' analytic modeling, auditing and performance measurements. Economic aspects on service computing are of interest for both service providers and service users.

Although several advantages of service computing are claimed, formal models, frameworks or tools to quantify its economic benefits (e.g., for service providers and service users) are not yet available. Several questions are still unanswered, e.g.: How can services be assessed (from an economic perspective)? How to predict services' performance? Which are the most relevant economic theories and approaches to support the formal definition and (economic) performance assessment in service computing?

These issues are tackled in this article. The scope of this article is to present preliminary results of an on-going research project aiming at advancing research in the

E.M. Maximilien et al. (Eds): ICSOC 2010 Workshops, LNCS 6568, pp. 51–61, 2011.

area of economic performance assessment in service computing by developing an analytical model, conceptual framework and metrics useful for economic performance analysis. Issues addressed in this article pertain to:

- synthesized theories and approaches useful for service computing formal modeling and economic performance assessment;
- discuss research challenges on assessing performance in service computing.

The rest of this article is organized as follows. The next section briefly introduces the concept of service computing. Section three refers to ways for economic performance modeling and measurements in the context of service computing. An approach for service computing modeling is then presented. Current research challenges on the economics of service computing are discussed in Section four. The article concludes with a section addressing the needs for further research.

2 Service Computing: A Brief Overview

2.1 Introduction

Services represent autonomous and platform-independent computational entities, which can be described, published, discovered, and dynamically assembled to deploy distributed interoperable systems (e.g., [1], [2]). As emphasized in [3], service-oriented computing promotes the idea of assembling application components in a network of services to create dynamic business processes and agile applications that cross different geographically distributed computing platforms and organizations. The Service Oriented Architecture (SOA) allows services to communicate and exchange information between distributed systems; it provides means for service providers to offer services, and service users to discover services.

Web services are services that make use of the Internet as communication platform, and open Internet-based standards, such as: Simple Object Access Protocol (SOAP[1]) to exchange data; Web Service Description Language (WSDL[2]) to describe services; Business Process Execution Language for Web Services (BPEL4WS[3]) to specify business processes and interaction protocols [1][4]. Service providers can register their services in a public service registry using the Universal Description Discovery and Integration (UDDI[4]). Web services are currently regarded as the most promising service-oriented computing technology [5].

Transaction policies for service-oriented computing are discussed in [6]. The authors argue for the use of declarative policy assertions to advertise and match support for different transaction styles. A system support for transaction coupling models as the policy-based contracts guiding transactional business process execution is also advanced. The impact of using advanced transaction meta-models for Web services is

[1] SOAP, http://www.w3.org/TR/soap/
[2] WSDL, http://www.w3.org/TR/wsdl
[3] BPEL4WS, http://www.ibm.com/developerworks/library/specificaion/ws-bpel/
[4] UDDI, http://www.uddi.org and http://www.oasis-open.org/committees/uddi-spec

analyzed in [7]. A meta-model for defining arbitrary advanced transaction models is also introduced.

Service Level Agreements (SLAs) are signed between parties, as a service contract, in order to define services and set (performance) service parameters. Approaches to SLA support (e.g., insurance approach, provisioning approach, adaptive approach) are discussed in [8]. A business-aware Web service transaction model that allows expressing and blending business and quality of service (QoS) aware transactions based on business agreements from SLA and business functions is described in [9].

Research challenges in the context of service-oriented computing are described in [1] and a research roadmap for service oriented computing is presented in [2]. The authors launch four main research themes (regarded as architectural layers):

- *Service foundations* (e.g., service-oriented middleware backbone that realizes the runtime infrastructure for the SOA). Major research challenges in this area are: dynamically reconfigurable runtime architectures, end-to-end security solutions, infrastructure support for data process integration and semantically enhanced service discovery.
- *Service composition* refers to roles and functionality for aggregating various services into a single service. Composability analysis for replaceability, compatibility, and process performance; dynamic and adaptive processes; QoS-aware service composition; and business-driven automated compositions are indicated as critical research challenges in this area.
- *Service design and development*, where engineering of service applications, flexible gap-analysis techniques, service versioning and adaptability, and service governance are considered major research challenges.
- *Service management and monitoring*, where self-configurating, self-adapting, self-healing, self-protecting management services are pointed as major research challenges.

2.2 Related Work on Service Assessment and Monitoring

Research on service assessment and monitoring is mainly related to technical issues. The most common metrics associated to services are related to the quality of service (QoS). QoS represents a combination of different qualities or properties of a service, such as [10]: availability (that is the percentage of time a service is operating), security (which is related to the existence of an authentication mechanism offered by the service, confidentiality, data integrity, non-repudiation, and resilience to denial-of-service attacks), response time (e.g., the time a service takes to respond to a certain request), throughput (e.g., the rate to which a service can process requests). Such metrics are of interest for both service providers and service users. For service providers, for example, the values of such metrics are of interest when implementing priority-based admission mechanisms [11]. As emphasized in [2], approaches that attempt to calculate the QoS by collecting quality ratings from the users of the service and then combining them are not sufficient for deriving a reliable quality measure for

a service. Cloud computing[5] brings the promise of providing a QoS guaranteed dynamic computing environment (e.g., [13]).

SLA defines the QoS attributes and guarantees a service. Concerning SLA monitoring, several approaches have been developed. An automated and distributed SLA monitoring engine is presented in [14], and a SLA management system built upon business objectives in presented in [15]. As emphasized in [2] and [16], research activities need to focus on using QoS metrics for selecting services and establishing trust among business partners. This emphasizes the need to concentrate on defining, collecting and calculating specific metrics for service monitoring.

However, service monitoring and assessment should not focus only on measurements related to technical characteristics (e.g., service availability). Economic issues on services (e.g., cost of a service, attained payoff) should be also analyzed and quantified, since they represent critical indicators for service providers and service users.

3 On the Economics of Service Computing

3.1 Background

Research and development work on the economics of service computing is scarce. Economic aspects of a utility computing service[6] are analyzed in [18], focusing mainly on service pricing, resource flexing and costs related to preventive security measures. Although the model used is quite simple (e.g., only limited groups of costs have been considered, such as: maintenance and upgrading), the authors illustrate its use in business decisions.

A methodology (price-at-risk) that considers uncertainty in the pricing decision is presented in [17]. The proposed methodology is aimed at optimizing the expected Net Present Value (NPV), and a numerical example is described to support this methodology. The main weakness of this work is the fact that it is around the NPV, which is an indicator often considered unreliable since it exhibits anomalous behavior (e.g., [19]). The impact of service execution is analyzed from a business perspective in [20], where services execution is adjusted and optimized based on state business objectives.

The issue of economic performance assessment in the context of service computing is of utmost importance for service providers and service users. Adequate models and metrics need to be developed. As emphasized in [21], new approaches and models (e.g., for performance assessment) do not have to be completely different from existing paradigms and economic analysis, since certain measures do not necessarily deny earlier (traditional) approaches. As consequence, existing paradigms, frameworks, models and metrics should be synthesized and analyzed in the context of service computing.

---

[5] In [12], a cloud is defined as a type of parallel and distributed system consisting of a collection of inter-connected and virtualized computers which are dynamically provisioned and presented as aggregated computing resources based on SLAs set by negotiating parties.

[6] Utility computing services deliver information services when needed, e.g., customers pay for what they use [17].

3.2 State-of-the-Art on the Economics of Service Computing

An extensive literature survey has been conducted. The most relevant theories and approaches which can be relevant for the economic performance assessment in service computing were identified and are briefly described below.

Game Theory. Game theory is a branch of applied mathematics that analyzes players who choose different actions in an attempt to maximize their payoffs (e.g., [21]). A cooperative game allows the formation of coalitions: players join forces based on a binding agreement. Main fields of application are: economic systems, biology, philosophy, and computer systems. The modeling approach of game theory is highly relevant to the concept of service computing, e.g., monotonicity and super-additivity properties, Shaply value. In a service computing environment, coalitions comprise service provider(s), service client(s), and service broker(s), and their agreements are enforced in SLA(s).

Graph Theory. A (directed) graph is a set of objects called vertices (or nodes) connected by (directed) edges. A common definition of a graph (e.g., [22]) is:
$G = (V,E)$, where the sets V are vertices (or nodes), and E edges, respectively. Graph theory has been applied in network analysis, mathematics, and computer science. Graph theory is relevant for service computing analysis because organizations (e.g., service clients, providers or brokers) are the nodes of a graph, and the relationships established between organizations (e.g., stipulated in the SLA) are in fact the edges, to which different weights and directions can be associated.

Transaction Cost Economics (TCE). TCE (e.g., [23], [24]), is mainly used to explain economic issues when resources specificity plays a critical role. It is relevant for service computing modeling since it includes costs which are often ignored, e.g., search costs: costs associated with searching for an appropriate service provider.

Petri Nets. According to Murata [25], a Petri Net is a 5-tuple:

$PN = (P, T, F, W, Mo)$, where P is a finite set of places, T is a finite set of transactions, F is a set of arcs, W is a weight function, and Mo is the initial marking. Main fields of application of Petri nets are: computer science, network analysis, production control. In the context of service computing, service providers and service clients can be regarded as place nodes; transition nodes can be, for example, organizations initiating a certain service. Although applied in several domains, Petri nets have certain limitations, such as: the execution of Petri nets is nondeterministic; lack of compositionality; lack of locality.

Social networks. A social network [26] is a social structure comprised of actors (*nodes*) indicating the way by which they are connected (*ties*). Social networks have been applied to study how companies interact. In the context of service computing, social networks can provide ways for companies (e.g., service providers) to gather information, prevent competition, or conclude in setting prices. Social networks can provide general insight for relationship between service provider and clients during the negotiation and setting up of the SLA.

Decision Theory. Given a decision problem, decision theory makes use of probability theory to recommend optimal decisions, or an option that maximizes (expected) utility. Fields of application include: economics, artificial intelligence. According to [27], a decision problem presumes: an association of a set of outcomes with each action; a measure U of outcome value which assigns a utility $U(\omega)$ to each outcome $\omega \in \Omega$; a measure of the probability of outcomes conditional on actions $Pr(\omega/a)$ denoting the probability that outcome ω is obtained after action $a \in A$. Based on these elements, the expected utility $EU(a)$ is defined in [27] as the average utility of the outcomes associated with an alternative, weighting the utility of each outcome by the probability that the outcome results from the alternative:

$EU(a) = \int_{\Omega} U(\omega) \Pr(\omega / a) d\omega$. Decision theory is relevant for service computing, especially when decision makers (e.g., service providers) need to make forecasts (e.g., concerning resource utilization).

The above-mentioned theories and approaches bring certain advantages and limitations when applied to service computing economic analysis. Formal representation, advantages and limitations for each of these theories relative to service computing, are summarized in Table 1. Other approaches are also relevant. For instance, consumer theory and producer theory, imperfect competition theories can be applied for understanding the role of certain actors in the context of service computing. Microeconomics could be relevant for setting appropriate service pricing.

With these considerations, a graph theoretic approach has been developed to better understand a service computing environment, supporting services characterization and economic modeling. A service computing environment is regarded as a multi-directed graph, where the nodes are represented by the organizations which perform different roles, e.g., service provider, service client, service broker.

Definition 1. Service: S $\quad S = (N, E, T, M).$ (1)

S = service;
N = set of nodes (organizations) and
N is a 3-touple: $N = (S_P, S_C, S_B)$, where: S_P = Service provider; S_C = Service client; S_B = Service broker[7];
E = set of edges, representing directed lines connecting organizations in the business environment, which reflect of relationships between two nodes (e.g., between S_P and S_B, or S_P and S_C);
T = the time of analysis;
M = set of metrics to be monitored.

The environment in which different S_P, S_B, and S_C collaborate and compete constitutes the Service Computing Business Environment (SCBE).

[7] The concepts of S_P, S_C, S_B have been detailed in [28]. Accordingly, S_B are trusted parties that oblige S_P to respect privacy laws or industry best practices. A S_B keeps track of available S_P, and may add additional information, e.g., concerning reliability, trust-worthiness, QoS.

Table 1. Approaches to service computing modeling and formal definition[8]

Theory/ model/ approach	Definition and basic elements	Main strengths for service computing modeling and formal representation	Main limitations for service computing modeling and formal representation
1.Game theory	-A game is a function: $v:2^N \rightarrow R$ -Players; strategies; specification of payoffs for each strategy profile.	-Easy to define types of relationships among service client, provider, broker. -Shapley value -SLA negotiation - SLA supports coalition formation	-Lack of formal models, metrics and tools to assess and monitor economic parameters. -Shapley value can be used only in particular cases. -Several axioms and theorems in game theory assume that coalitions consist in disjunctive sets.
2.Graph theory	-Graph: $G = (V,E)$; V=nodes, E=edges.	-Service providers, service clients and service brokers and their relationships can be represented as a graph for analysis. -Algorithms	-Often not able to model real world situations.
3.TCE	Particular focus on transaction costs.	-Strategic costs related to SLA.	-Other costs involved in service computing are not included.
4.Petri nets	-5-Tuple (P, T, F, W, Mo, K); P=places; T=transitions; F=flow relation; Mo=initial marking; W=arc weights; K=capacity restrictions.	-A service computing environment can be represented as a Petri net for certain analyses.	-Lack of locality and compositionality
5.Social networks	-Nodes -Ties	-Relationships among service providers, clients, brokers).	-Extensive focus on social aspects among nodes.
6. Decision theory	- ω-outcome; U(ω)-utility; Pr(ω/a); a-action; EU-expected utility: $EU(a)=\int_{\Omega} U(\omega)\Pr(\omega/a)d\omega$.	-Forecasting (e.g., resource use of service providers).	-Often difficult to obtain reasonable estimates.

Figure 1 succinctly portraits these concepts. The SCBE is formed by 9 entities (e.g., with the role of S_P, S_B, and S_C). Similar to graph theory, entities are represented as nodes, which are directed edges reflecting specific business relationships among them. The relationships between entities are reflected in the SLA and correspond to the observation time, T. Some service providers (e.g., S_{P4}) may not have any service client. Different from game theory where coalitions comprise disjunctive sets, this view on business relationships states that any entity (e.g., S_P, S_B, S_C) can have different types of relationships with other organization(s) (e.g., service providers may offer to the same client different types of services, and each business relationship is reflected in a different SLA).

[8] Following the approach presented in [21].

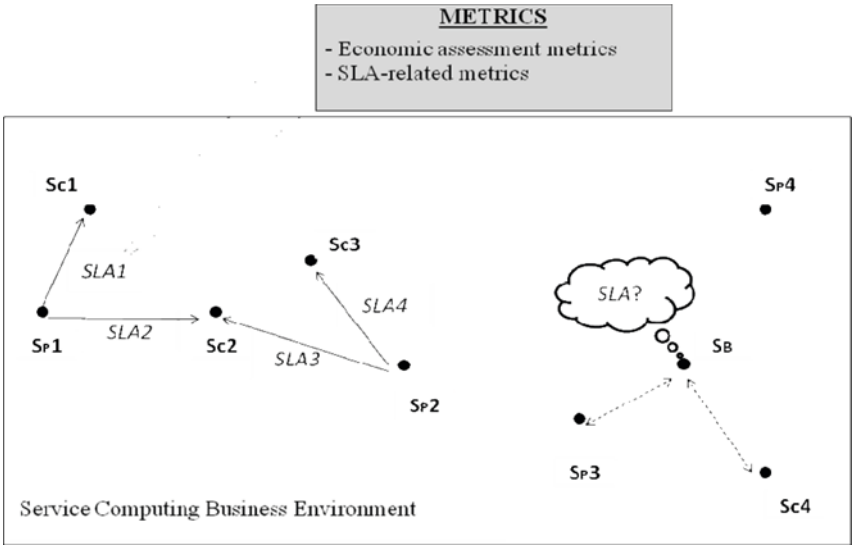


Fig. 1. Pictorial representation of a service business environment

To assess services, metrics related to SLA should be calculated and monitored (e.g., availability, throughput, downtime, response time), as well as metrics for assessing the economic performance (e.g., costs, payoffs, ROI). Models, metrics, methodologies, frameworks and tools for supporting an economic analysis and assessment in a service computing environment are not yet available. However, they would be of utmost importance for SLA negotiation, set up and monitoring, and to support decision making for S_C and S_P (e.g., concerning S_P resource scheduling).

3.3 Economic (Performance) Assessment in the Service Computing Context

In a service computing environment, S_P, S_B, and S_C collaborate and compete, and the relationships between them are characterized by several attributes, some quantifiable (e.g., costs associated with SLA negotiation), while others are difficult to be quantified.

In [21], three main ways to assess the (economic) performance have been emphasized: performance indicators, benchmarking methods, and frameworks. Performance indicators represent quantitative measures. In the context of service computing, performance indicators could support the measurement of specific characteristics related to services, organizations and their relationships, such as: costs (e.g., costs associated with SLA negotiation); QoS metrics; payoffs attained by outsourcing a service instead of in-house development. Benchmarking models can be applied to determine how well an entity (e.g., S_P) is performing compared to others. All four types of benchmarking methods (internal; competitive; functional; generic) can be applied in the context of service computing. Benchmarking assumes there is always an opportunity for improving current situation and the existence of a statistically meaningful sample size, e.g., [29], [21]. Several frameworks for assessing performance have been developed, e.g., Balanced Scorecard [30]. As emphasized in [21], the main limitation of benchmarking methods and frameworks for performance assessment is their total dependence on indicators as reference for analysis and comparison.

Thus, it is critical to develop appropriate metrics to assess performance in the context of service computing. Examples of such metrics are briefly introduced below.

Costs. The costs incurred in a SCBE are the costs associated with all activities related to providing or outsourcing a service. For *service clients*, costs to be considered include: *strategic costs* (e.g., cost of outsourcing the service); *negotiation costs* (e.g., SLA negotiation and re-negotiation costs: a S_C may simultaneously negotiate with different S_P before setting up a SLA with one S_P); *search costs* (e.g., costs incurred in determining if a certain service is available on the market); *legal costs* (e.g., costs due to legal actions when the SLA has not been respected); *costs associated to risks*. For *service providers*, costs may include, among others, *costs associated to service development*; *costs associated to maintenance* (e.g., yearly software upgrade costs); *management costs* (e.g., scheduling of resources).

SLA monitoring metrics, e.g., QoS: availability, throughput, response time.

4 Research Challenges

This section is focusing on research challenges on (economic) performance assessment in the context of service computing.

Analytical Modeling and Economic Performance Assessment and Monitoring. Performance assessment measurements refer to the quantification of performance-related characteristics of services. Such metrics should be calculated in an automatic way. Research in this area should focus on defining models, methodologies, frameworks, metrics and tools to measure (economic) performance characteristics in the context of service computing, e.g., costs. As emphasized in [31], practitioners have almost no support in selecting appropriate metrics for the implementation of SLAs, e.g., concerning automation and compliance with service objectives and IT management processes. Such metrics and tools are useful in many ways, e.g., to identify abnormalities (e.g., to monitor SLA and identify SLA violations). The challenge is not only to define performance metrics, but also to calculate and monitor them in an automated manner, and elaborate *correlations* between these metrics. Such correlations could support, for example, optimization, and identification of its impact, e.g., on S_P resource scheduling decisions.

Risk Analysis. Risk analysis is useful to analyze the effectiveness of resources management policies in achieving the objectives set. Separate risk analysis (e.g., the analysis of a single objective for a particular scenario) and integrated risk analysis (e.g., analysis of multiple objectives) approaches should be performed in the context of service computing.

Interoperability. Interoperability is of utmost importance in service computing for both service providers and service clients, e.g., for enterprise outsourcing a service is critical to attain seamless interoperability between the software/ services outsourced and own software. Research on interoperability should focus not only on technical and information interoperability, but also on economic/ business interoperability aspects.

Holistic Approach. Research on service computing should combine different areas, e.g., computer science/ engineering, economics, mathematics.

5 Conclusions and Future Research Work

In order to achieve their business goals, several enterprises decided to outsource services and focus on their core areas. In a service oriented architecture, services represent self-contained modules that provide business functionalities and are independent of the state and context of other services [28]. Several research and development projects are currently being pursued focusing on service computing. However, most of existing research work is focusing on technical aspects. Research on the economics of service computing is scarce. Adequate analytical methods and tools to quantify the economic benefits of service computing need to be developed.

Economic performance assessment in the context of service computing is a challenge. Metrics and tools to support such an analysis are necessary to assess current (economic) performance, and forecast performance capabilities.

In this article, the results of a brief survey have been presented in Table 1, with the main theories and approaches relevant for service modeling and assessment, e.g., game theory, graph theory, decision theory, Petri nets, transaction cost economics. Their strengths and limitations in the context of service computing have been analyzed. Based on this review and the research pursued so far, types of costs for both service providers and service clients have been identified. Research challenges in this area have been succinctly discussed.

Future research will focus on developing economic performance indicators for service computing assessment, and their validation.

References

1. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F.: Service-Oriented Computing: State of the Art and Research Challenges. IEEE Computer, 64–70 (June 2007)
2. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F.: Service-Oriented Computing: A Research Roadmap. International Journal of Cooperative Information Systems 17(2), 223–255 (2008)
3. Leymann, F.: Combining Web Services and the Grid: Towards Adaptive Enterprise Applications. In: Proceedings CAiSE 2005 Workshops. FEUP Edições, vol. 2, pp. 9–21 (2005)
4. Papazoglou, M.P.: Web Services: Principles and Technology. Prentice-Hall, Englewood Cliffs (2007)
5. Weerawarana, S., et al.: Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More. Prentice Hall, Englewood Cliffs (2005)
6. Tai, S., Mikalsen, T., Wohlstadter, E., Desai, N., Rouvellou, I.: Transaction policies for service-oriented computing. Data & Knowledge Engineering 51, 59–79 (2004)
7. Hrastnik, P., Winiwarter, W.: Using advanced transaction meta-models for creating transaction-aware Web service environments. International Journal of Web Information Systems 1(2), 89–99 (2006)
8. Verma, D.C.: Service Level Agreements on IP Networks. Proceedings of the IEEE 92(9), 1382–1388 (2004)
9. Papazoglou, M.P., Kratz, B.: A Business-Aware Web Service Transaction Model. In: Dan, A., Lamersdorf, W. (eds.) ICSOC 2006. LNCS, vol. 4294, pp. 352–364. Springer, Heidelberg (2006)

10. Menascé, D.A.: QoS Issues in Web Services. IEEE Internet Computing, 72–75 (November-December 2002)
11. Cherkasova, L., Phaal, P.: Session-based Admission Control: A Mechanism for Peak Load Managementof Commercial Web Sites. IEEE Transactions Computers 51(6), 669–685 (2002)
12. Buyya, R., Yeo, C.S., Venugopal, S., Broberg, J., Brandic, I.: Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation Computer Systems 25, 599–616 (2009)
13. Wang, L., von Laszewski, G., Younge, A., He, X.: Cloud Computing: A Perspective Study. New Generation Computing 28, 137–146 (2010)
14. Sahai, A., Machiraju, V., Sayal, M., van Moorsel, A.P.A., Casati, F.: Automated SLA Monitoring for Web Services. In: Feridun, M., Kropf, P.G., Babin, G. (eds.) DSOM 2002. LNCS, vol. 2506, pp. 28–41. Springer, Heidelberg (2002)
15. Buco, M.J., Chang, R.N., Luan, L.Z., Ward, C., Wolf, J.L., Yu, P.S.: Utility computing SLA management based upon business objectives. IBM Systems Journal 4(1), 159–178 (2004)
16. Maximillien, E.M., Singh, M.P.: Toward Automatic Web Services trust and Selection. In: Proceedings of 2nd International Conference on Service Oriented Computing (ICSOC), pp. 212–221. ACM Press, New York (2004)
17. Paleologo, G.A.: Price-at-Risk: A methodology for pricing utility computing services. IBM Systems Journal 43(1), 20–31 (2004)
18. Degabriele, J.P., Pym, D.: Economic aspects of a utility computing service. HPL-2007-101 Technical Report (2010), http://www.hpl.hp.com/techreports/2007/HPL-2007-101.html (accessed on September 30, 2010)
19. Oehmke, J.F.: Anomalies in Net Present Value Calculations. Economics Letters 67, 349–351 (2000)
20. Casati, F., Shan, E., Dayal, U., Shan, M.-C.: Business-oriented management of Web services. Communications of ACM 46(10), 55–60 (2003)
21. Chituc, C.-M., Nof, S.Y.: The Join/ Leave/ Remain (JLR) decision in collaborative networked organizations. Computers & Industrial Engineering 53, 173–195 (2007)
22. Gross, J.L., Yellen, J.: Handbook of graph theory. CRC Press, Boca Raton (1999)
23. Williamson, O.E.: Transaction-cost economics: The governance of contractual relations. Journal of Law and Economics 22(2), 233–261 (1979)
24. Williamson, O.E.: Transaction cost economics. Handbook of New Institutional Economics, pp. 41–65. Springer, Dordrecht (2005)
25. Murata, T.: Petri nets: Properties, analysis and applications. Proceedings of IEEE 77, 541–580 (1989)
26. Barnes, J.A.: Class and committees in a Norwegian Island Parish. Human Relations (2954)
27. Doyle, J., Thomason, R.H.: Background to Qualitative Decision Theory. AI Magazine 20(2), 55–68 (1999)
28. Papazoglou, M.P., van den Heuven, W.-J.: Service oriented architectures: approaches, technologies and research issues. The VLDB Journal 16, 389–415 (2007)
29. Brewer, P.: Putting strategy into balanced scorecard. International Federation for Accountants (2003), http://www.ifac.org
30. Kaplan, R., Norton, D.: The Balanced Scorecard – translating strategy into action. Harvard Business School Press, Boston (1996)
31. Paschke, A., Schnappinger-Gerull, E.: A Categorization Scheme for SLA metrics. In: Proceedings MKWI 2006, Passau, Germany, pp. 25–40 (2006)

Engineering Service-Oriented Applications 6th International Workshop WESOA 2010

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

[1] Karlsruhe Institute of Technology
Christian.Zirpins@kit.edu
[2] Prague University of Economics
jirif@vse.cz
[3] University of Technology, Sydney
george.feuerlicht@uts.edu.au
[4] University of Hamburg
lamersdorf@informatik.unihamburg.de
[5] University of Extremadura
gobellot@unex.es

Abstract. Many of today's large-scale software projects in the area of distributed systems and especially enterprise IT adopt service-oriented software architecture and technologies. For these projects, availability of sound software engineering principles, methodology and tool support is of utmost importance. However, traditional software engineering approaches are not fully appropriate for the development of service-oriented applications. The limitations of traditional methods in the context of service-oriented computing have led to the emergence of Software Service Engineering (SSE) as a specialist discipline, but research in this area is still immature and many open issues remain. The WESOA workshop series brings together research community and industry practitioners in order to develop comprehensive engineering principles, methodologies and tool support for the entire software development lifecycle (SDLC) of service-oriented applications.

Keywords: Service Computing, Software Engineering.

1 Aims and Scope

The WESOA series addresses challenges of SSE that arise from specific characteristics of service-oriented applications that are often process-driven, loosely coupled, composed from autonomous services within complex IT landscapes and closely related to diverse socio-economic contexts. Service-oriented Computing (SOC) enables the materialization of organizational processes as flexible compositions of autonomous service components. Stakeholders, domain experts, software architects and engineers instrument Service-Oriented Architectures (SOA) to drive constant organizational change by means of agile reengineering of software services, system landscapes and applications. In particular, service-oriented applications need to provide multiple, flexible and sometimes situational interaction channels within and beyond

E.M. Maximilien et al. (Eds): ICSOC 2010 Workshops, LNCS 6568, pp. 62–64, 2011.

organizational structures and processes. Engineering of such software systems requires continuous, collaborative and cross-disciplinary development processes, methodologies and tools that synchronize multiple SDLCs of various SOA artifacts with organizational innovation processes.

Service-oriented applications closely resemble the organizational principles of their application domains that are often dynamic, collaborative and process-driven networks. They are compositions of service system components that are provided by autonomous stakeholders based on unique assets and capabilities. Thus, SSE is a complex undertaking that spans multiple distributed yet collaborative development processes of interrelated SOA components. Additionally, service-oriented applications often have a socio-economic as well as political dimension. It is the challenge of SSE not only to cope with these specific circumstances but also to capitalize on them with radically new approaches. The WESOA series addresses these challenges and particularly concentrates on principles, methodologies and tool support that consider the holistic scope of the service-oriented SDLC.

2 Workshop Event

The 6th WESOA event was held in San Francisco on December 7, 2010. The workshop opened with a keynote presentation by Greg Pavlik from Oracle giving an industry perspective on SOA platforms, standards, and applications and exploring current trends in service-oriented computing.

The technical sessions consisted of eight high-quality papers representing a rich variety of topics revolving around principles, methods and application of SSE.

The first session included contributions by Ian Gorton et al. on *Engineering High Performance Service-Oriented Pipeline Applications with MeDICi*, Rosa Alarcon et al. on *Hypermedia-driven RESTful service composition*, and Lukasz Juszczyk et al. on *CAGE: Customizable Large-scale SOA Testbeds in the Cloud.*

The next session included contributions by Soudip Roy Chowdhury et al. on *Wisdom-Aware Computing: On the Interactive Recommendation of Composition Knowledge*, Marcus Roy et al. on *Facilitating Enterprise Service Discovery for Non-Technical Business Users*, and Thomas Heinze et al. on *Process Restructuring in the Presence of Message-Dependent Variables.*

The final session included contributions by George Feuerlicht on a *Simple Metric for Assessing the Quality of Service Design*, and Seyed Hossein Siadat, et al. on *Adaptation of Web Services based on QoS Satisfaction*.

3 Workshop Co-organizers

- Christian Zirpins, Karlsruhe Institute of Technology, Germany
- George Feuerlicht, Prague University of Economics, Czech Republic
- Winfried Lamersdorf, University of Hamburg, Germany
- Guadalupe Ortiz, University of Extremadura, Spain

4 Program Committee

- Sudhir Agarwal, KIT Karlsruhe
- Marco Aiello, University of Groningen
- Djamal Benslimane, LIRIS
- Sami Bhiri, DERI Galway
- Alena Buchalcevova, Prague Uni. of Economics
- Robert Castaneda, CustomWare
- Anis Charfi, SAP Research Center
- Jen-Yao Chung, IBM T.J. Watson Research
- Oscar Corcho, Univ. Politécnica de Madrid
- Vincenzo D'andrea, University of Trento
- Florian Daniel, University of Trento
- Valeria de Castro, University Rey Juan Carlos
- Schahram Dustdar, Technical University Vienna
- Keith Duddy, Queensland Univ. of Technology
- Howard Foster, Imperial College London
- Paul Greenfield, CSIRO
- Birgit Hofreiter, Hochschule Lichtenstein
- Dimka Karastoyanova, University of Stuttgart
- Rannia Khalaf, IBM Research
- Agnes Koschmieder, KIT Karlsruhe
- Winfried Lamersdorf, University of Hamburg
- Mark Little, Arjuna,
- Tiziana Margaria, University of Potsdam
- E. Michael Maximilien, IBM Research
- Massimo Mecella, University Roma La Sapienza
- Daniel Moldt, University of Hamburg
- Rebecca Parsons, ThoughtWorks
- Cesare Pautasso, University of Lugano
- Greg Pavlik, Oracle
- Pierluigi Plebani, Politecnico di Milano
- Franco Raimondi, University College London
- Wolfgang Reisig, Humboldt-University Berlin
- Norbert Ritter, University of Hamburg
- Colette Rolland, University of Paris
- Jim Webber, ThoughtWorks
- Willem-Jan van den Heuvel, Tilburg University
- Olaf Zimmermann, IBM Research Zürich

Acknowledgements

The WESOA 2010 organizers 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 to the workshop chairs Gustavo Rossi, Soe-Tsyr Yuan, and Michael Maximilien for their direction and guidance.

Guadalupe Ortiz acknowledges the support from *Ministerio de Ciencia e Innovación* (TIN2008-02985).

Adaptation of Web Services Based on QoS Satisfaction

Barbara Pernici and S. Hossein Siadat

Politecnico di Milano, Italy
{pernici,siadat}@elet.polimi.it

Abstract. Requirements of Service Based Applications (SBAs) tend to change during the life cycle of the service. Therefore, adaptation and evolution of services become a necessity in order to provide the agreed Quality of Service (QoS) stated in a contract between the service provider and requestor. Recently, many adaptation methods have been proposed in the literature. However, there is no overall consensus in selecting the best strategy and the consequences of adaptation are usually neglected. In this paper, we propose an approach for service adaptation through defining a flexible service description using fuzzy parameters. This approach provides a compatibility mechanism that measures the aggregated satisfaction value of offered services to understand to what extent the quality changes are satisfiable according to the existing contract. According to the degree of satisfaction function we then propose our adaptation/evolution strategy.

1 Introduction

With the rapid propagation of Service Based Applications (SBAs), evolutionary changes of Web Services are gaining huge interest among the practitioners. Changes are due to several reasons and may occur continuously over time. Such changes need to be captured and analyzed as they have various requirements, causes and effects. These changes then will be compared with Service Level Agreement (SLA) which is a guarantee of a certain agreement between the service provider and the requestor of the service. This paper is concerned with non-functional requirements of services particularly QoS issues, and we investigate QoS changes through a contractual approach. Therefore, we define a fault/violation when a service is not delivered according to the predefined quality description in the contract. Typical non-functional properties include availability, response time, cost and throughput are often collectively referred to as quality dimensions. While [18] provides a general categorization of SLA metrics, [24], [21] and [19] present a description of QoS attributes and frameworks for Web service selection and discovery. It is possible to define QoS description as an XML specification of SLAs in several languages such as Web Service Level Agreement (WSLA) [10] by IBM or Web Service Management Language (WSML) [20] by HP and WS-Agreement [3].

E.M. Maximilien et al. (Eds): ICSOC 2010 Workshops, LNCS 6568, pp. 65–75, 2011.

Deviation of quality ranges from the existing contract may produce a system failure and bring dissatisfaction for customers. To this end, the evolution and adaptation of web services are becoming two important issues in reacting to the various changes in order to provide the agreed QoS stated in the contract. Recently, many adaptation strategies and methods have been proposed in the literature. A list of adaptation strategies for repair processes in SBAs is provided in [6] and [1].

However, there is no overall consensus in defining and selecting the best strategy when a change occurs. Furthermore, business aspects like cost, the consequences of adaptation actions, say service replacement, and customer satisfaction are usually neglected. This is of great importance to consider the cost of selecting another service provider and establish a new contract which could be time consuming as well in some circumstances. Taking into account that, in many cases selecting a new service provider as an adaptation action is a costly decision without significant achievement with respect to QoS. Therefore, any service replacement needs to be analyzed specially when there is a high degree of loyalty between the involving parties in long term contracts.

We address these limitations by defining a service satisfaction degree that allows us to select an appropriate adaptation action. The research challenge we are following in the paper is twofold. Firstly, how to provide compatibility between the provider and the requestor according to the existing contract once the QoS is changed? Second, how to define a mechanism for selecting an appropriate adaptation strategy. In order to address these research challenges a few steps are required.

1. Defining the service description in a flexible approach to describe quality of the provided service. For this, we extend the service description defined in [4] by using a fuzzy approach for defining quality dimensions. This way we understand to what extent a quality parameter is satisfied/violated. Quality parameters are usually defined in a strict and non-flexible manner. Even when they were defined within ranges, any value out of the range is considered to be a violation. Therefore, a monitoring engine behaves in a binary approach which results in either violation or not. In this paper, we consider a fuzzy approach for measuring violation of each parameter. Therefore, we are able to measure the degree of violation/satisfaction of each quality parameter.
2. Providing a satisfaction function taking advantages of the service description equipped with fuzzy parameters. According to the satisfaction degree gained from the satisfaction we then propose an adaptation strategy to be performed. In the following, we start by a motivating example in Section 2. Section 3 introduces the major related work. In Section 4 we show the formalization of service description through defining fuzzy parameters and in Section 5 we define a satisfaction function to measure to what extend the QoS is achieved according to the existing contract. We then define the extended compatibility mechanism and adaptation algorithm in Section 6 that work based on the result of the satisfaction function. Adaptation strategies will be discussed in Section 7 and we conclude the paper in Section 8.

2 Motivating Example

Here we provide a motivating example to illustrate the service adaptation problem. Suppose that there is an airline agency providing plane tickets for its customers. It provides different services offered by various airlines. Consider that each service is associated with three parameters of Availability, Response Time and Cost (other aspects i.e. flight time and number of stopovers are applicable). Customers show a rather relaxed behavior for the availability and response time of the services, but they are very strict with the Cost. Underlying this example there is the following assumption: Customers have strong commitment for their provider, therefore they are interested in using services from the same provider.

Suppose that a service is found and a contract is negotiated between a provider and a customer such that it offers a flight from location A to B with defined quality parameters. The quality of the offered service can dynamically change during the execution. For example, the service provider has a delay in providing the offered service. In such cases, an adaptation strategy needs to be performed in order to comply the violated situation. We consider two main levels of adaptation actions. The first and the easiest one is to choose another service provider that offers a service with the same functionality (i.e. providing a flight from location A to B), however the new service probably has different QoS guarantees. This approach is interpreted as service replacement. The second option is to renegotiate internally between the same provider and customer. For example the violation could be compensated by offering a lower price for the service. Before taking any action, some criteria need to be taken into account and from different perspectives. A new service provider may offer a service with a better quality, however the cost and consequences of choosing another provider need to be evaluated. This includes the cost of extra queries need to be done to find the best new provider and also the cost of a new contract that should be established. There are other issues involved from the customer perspective, for example he may have preferences to remain with the current provider due to the loyalty aspects. However, the internal renegotiation may not be possible all the time if the violation level leads to the dissatisfaction of customers (i.e. if the offered service exceeds the maximum budget). Considering all the aforementioned points, taking an option with best quality offered is not the ideal solution. There is a need to trade-off between the overall value of the offered quality and the cost factor. Fig. 1 shows an overall model of the approach.

3 Related Work

Recently, service adaptation and evolution in SBAs have gained increasing attention among the practitioners and researchers. Chafle et al. [7] presented A-WCSE approach for adaptive web service composition and execution. The approach allows adaptation by generating multiple workflows at different stages. Canfora et al. [5] presented an approach for QoS-aware replanning of service composition at run time using late-binding technique. As a result there is much previous work

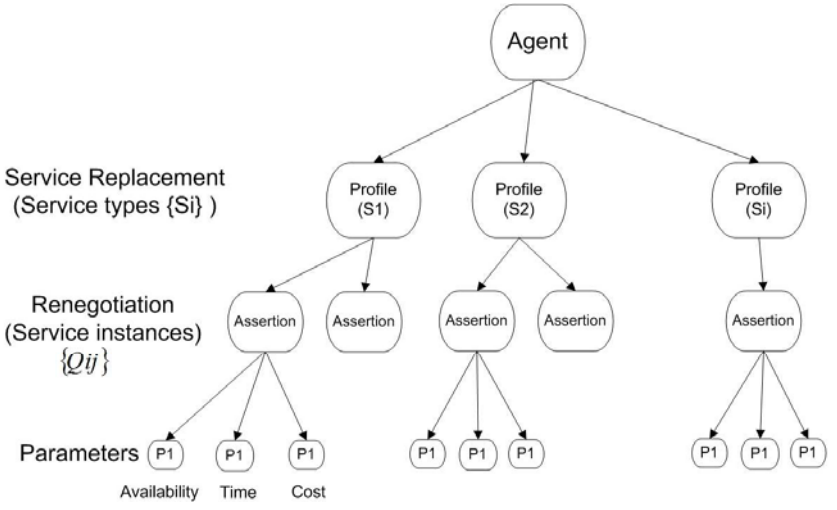


Fig. 1. Motivating Example (Overall approach)

on adaptation strategies and techniques. However, there is no overall consensus in selecting the best strategy and the consequences of such adaptation strategies are usually neglected. For this purpose some research has been investigated to consider the overall value of a change. Harney and Doshi [8] presented a mechanism called *Value of changed information* (VOC) to estimate the overall value brought by a change. He et al. [9] has applied the VOC mechanism for the adaptation of web service based on workflow patterns. None of the above approaches consider a satisfaction level for the offered service.

We argue that the problem lies in defining a flexible description for Web Services. Although a lot of research has been conducted for functional Web Service description, only a few efforts have been done with respect to non-functional properties description of Web Service. Among syntactic and semantic WS description we refer to the works done by [15] and [11] which they also provided algorithms for service selection based on such description. A major limitation of these works and other similar ones is due to not considering the partial satisfaction of the QoS attributes. With this regards, [17] provided a semantic Policy Centered Meta-model (PCM) approach for non-functional property description of web services. A number of operators (e.g. greaterEqual, atLeast) for numeric values are defined in the model for determining tradeoffs between various requests. Therefore, the approach can support the selection of Web Services that partially satisfy user constraints. In [13] and [12], the authors extend the approach proposed in [17] by proving a solution for Web Service evaluation based on constraint satisfaction problem. The approach uses utility function to present the level of preferences for each value ranges defined in the service description. However, is does not take care of adaptation issues and controlling values at run-time. In [16], the authors discuss about fixable and non-fixable properties

to deal with bounded uncertainty issue. Constraint programming is used as a solution for the said issue, however, there is no evaluation of the work.

Fuzzy approach [23] has been used in several systems and application to perform the fuzzy behavior. [14] defined a set of fuzzy requirements in software development. We used fuzzy parameters in defining service description to evaluate the satisfaction degree of each parameter and to perform an appropriate adaptation action based on the satisfaction level.

4 Fuzzy Parameters for Quality Dimensions

This section is devoted to demonstrate the formal definitions of *parameters*, *profiles* and *assertions* in a service description. The formal specification we propose has been inspired and is an extension of our previous work [4]. We extended the work by taking advantage of fuzzy approach providing two kinds of parameters namely *fuzzy* and *non-fuzzy*. Having introduced the fuzzy parameters it is possible to understand to what extent the quality parameters are violated/satisfied.

In order to formally define fuzzy parameters, we need to introduce some background from [4]. We define set $\mathcal{D}$ to contain the quality dimensions (such as availability, execution time, price or throughput) identified and agreed by the service provider and consumer. Each quality dimension has a domain and range; e.g., availability is a probability usually expressed as a percentage in the range 0-100% and execution time is in the domain of real numbers in the range $0..+\infty$. A quality dimension d can be considered *monotonic* (denoted by d^+) or *antitonic* (d^-); monotonicity indicates that values closer to the upper bound of the range are considered "better", whilst with antitonic dimensions values closer to the lower bound are considered better. A *parameter* m associates a quality dimension to a value range.

If a parameter is non-fuzzy (strict), its satisfaction degree will be evaluated in a binary manner (which is a Yes or No). In contrast, fuzzy parameters (relaxed) will be evaluated in a fuzzy manner which shows different degree of satisfaction ($x \in [0, 1]$). Note that we also provide value ranges for both parameters regardless of being fuzzy or non-fuzzy. The degree of satisfaction will be evaluated using a satisfaction function we introduce in the next section. In the following we provide the extended definition of a *parameter* based on the definition introduced in [4].

Definition 1 (Parameter). *We define a Parameter* $m \in \mathcal{M}$ *as a tuple* $m := (d, v, f), d \in \mathcal{D}, v \in \mathcal{V}, f \in \{s, r\}$. *where* $\mathcal{D}$ *is the set of quality dimensions,* $\mathcal{V}$ *is the set of ranges for all quality dimensions* $\mathcal{D}$*, s represent a strict parameter and r represent a relaxed parameter.*

Having defined the quality dimensions and service description, a contract will be defined based on a matching and mapping functions defined in our previous work [4] in a similar manner. We do not present the functions here as they are not the concern of this paper.

The definition of *assertion* is the same as in [4]. Assertions can be combined in a similar fashion to form a *profile*. For the sake of simplicity we also skip the definitions of *assertion* and *profile*.

Table 1. Working example: formal specifications

Provider $\mathcal{P}$	$l_p = (q_{p1}, q_{p2})$
	$q_{p1} = (m_{p1,1}, m_{p2,1}, \overline{m_{p3,1}})$ $m_{p1,1} = (d_1, [0.8, 0.95]), m_{p2,1} = (d_2, [2, 5]), \overline{m_{p3,1}} = (d_3, [1, 1])$
	$q_{p2} = (m_{p1,2}, m_{p2,2}, \overline{m_{p3,2}})$ $m_{p1,2} = (d_1, [0.7, 0.95]), m_{p2,2} = (d_2, [3, 5]), \overline{m_{p3,2}} = (d_3, [0.8, 0.8])$
Consumer $\mathcal{C}$	$l_c = (q_{c1})$
	$q_{c1} = (\overline{m_{c1,1}}, \overline{m_{c2,1}}, m_{c3,1})$ $\overline{m_{c1,1}} = (d_1, [0.8, 0.9]),\ \overline{m_{c2,1}} = (d_2, [3, 5]),\ m_{c3,1} = (d_3, [0, 1])$
Dimensions $\mathcal{D}$	$d_1^+ =$ `Availability`, $d_2^- =$ `ResponseTime`, $d_3^- =$ `Price`

During a service life-cycle, as the application runs, QoS offerings may change due to several reasons. Therefore, adaptation of web services needs to be performed in an appropriate manner to accommodate QoS changes/violations by choosing the best adaptation strategy. Defining service description with the aforementioned fuzzy parameters provides a more flexible situation dealing with adaptation decisions. We discuss how it can facilitate the adaptation of web services through our motivating example according to Table 1. According to the new definition of parameters we already explained, let us assume that Availability and ResponseTime are fuzzy parameters and Cost is a non-fuzzy Parameter.

In [4] we provided situations in which new QoS ranges could be still acceptable for both parties according to the existing contract. We defined a compatibility mechanism that uses parameter subtyping and used Allen's Interval Algebra [2] in order to express the subtyping. The provider and requestor are compatible with each other according to the existing contract if the QoS changes are in one of the acceptable situations. If the compatibility is not provided, however it does not give any information about the degree of satisfaction/dissatisfaction of the offered service. Let us consider an availability of 80 to 90%. For example if the new range of availability is less than 80%, this is not considered as an acceptable situation and it is considered as a violation. In such cases, we would also like to understand to what extent the quality parameter and the aggregated service quality are satisfactory. An availability of 75% might still be acceptable if we consider a general value of all parameters. In the following we will concentrate on the definition of a more precise compatibility mechanism taking advantage of the fuzzy approach we have already presented. To this end, we introduce a satisfaction function in the next section.

5 Satisfaction Function

We associate a satisfaction value of $S \in [0, 1]$ to each parameter. In order to calculate the satisfaction value $A(m)$ for a given parameter m, we use Equations 1 and 2 for monotonic and antitonic parameters respectively as follow:

$$A(m) = \begin{cases} m' \div m_{min} & \text{InCompatible} \\ 1 & \text{Compatible} \end{cases} \quad (1)$$

Note: Equation 1 and 2 are only applicable for fuzzy parameters while non-fuzzy parameters receive a satisfaction value of 0 in case of incompatibility.

For monotonic dimensions: the range between 0 and the minimum range in the contract will be mapped to [0, 1]. This is actually the area of violation according to the contract (reported as negative weight in some work). Any value between the min and max is considered to be compatible (satisfaction value of 1). For antitonic dimensions, the range between 0 and maximum value in the contract is considered to be compatible (the satisfaction value 1) and the range between maximum value and the actual new value is mapped to the satisfaction value of [0, 1].

$$A(m) = \begin{cases} m_{max} \div m' & \text{InCompatible} \\ 1 & \text{Compatible} \end{cases} \tag{2}$$

In order to calculate the overall satisfaction degree of the service including satisfaction value of all the parameters, we can apply the following method in Equation 3. Consider $A(Q)$ as the satisfaction of the assertion Q that has a set of parameters. Since the parameters are structured in an AND-refined approach with s strict and f fuzzy parameters, therefore, the overall satisfaction value could be measured as below:

$$A(Q) = (A(Mi) \wedge ... \wedge A(Ms)) * \min(A(Mj), ...A(Mf)) \tag{3}$$

$A(Mi) = 0, 1 \quad i = 1, ..., s$ (satisfaction value of strict parameters)
$A(Mj) \in [0, 1] \quad j = 1, ..., f$ (satisfaction value of fuzzy parameters)

It can be interpreted from Equation 3 that the overall satisfaction of the service depends on the minimum satisfaction of its fuzzy parameters. The offered QoS will not be satisfied if at least one of the strict parameters is not satisfied. For example if the cost is more than the maximum amount, the overall satisfaction value is 0. As for the fuzzy parameters, let us assume that the perceived availability is 70% (the range in the contract is between 80% to 90%), and the response time is 3s (compatible according to range in the contract). Therefore, according to Equation 1 the satisfaction value is 0.88 and according to Equation 3 the overall service satisfaction is 0.88. Now, let us consider the case that the response time is 6s, then according to the Equation 2 the satisfaction value of time parameter is 0.83 and therefore, the overall service satisfaction is 0.83.

The satisfaction value could be compared with the satisfaction value of other assertions of the current web service or with the value of other web services in the composition. The existing contract will remain valid between the service provider and the customer as long as the satisfaction value is more than any other services in the composition. While the contract is not valid when the satisfaction value is less than an agreed value or a service provider with a better value is found. A service replacement may occur in this situation. Besides from the satisfaction factor, we introduce a *loyalty* degree which is a degree of commitment between a provider and consumer. The service loyalty/fidelity degree is taken into account before any decision for adaptation. This is due to the fact that both customer and service provider (involving parties) want to remain with the existing contract

(specially for the long term contracts). We therefore define a total satisfaction degree using the loyalty degree as below in Equation 4:

$$A_t(Q) = A(Q) * LoyaltyDegree \tag{4}$$

Let us consider a loyalty degree of 1.1 between the customer and service provider, therefore, the overall satisfaction value of the previous example will be 0.91 (0.83*1.1). An initial degree of loyalty could be given to a provider and specified in the contract by the customer. In the next section we provide an algorithm for dealing with adaptation/evolution decisions.

6 Compatibility Mechanism

This chapter presents an algorithm for choosing the best adaptation/evolution strategy once a QoS violation occurs. The algorithm is based on the mechanism presented in Section 5 and the compatibility mechanism we introduced in [4]. Fig. 2 shows the pseudo code for selecting adaptation actions based on the QoS satisfaction value of the offered service.

```
Algorithm for choosing Adaptation Strategies
Input S1 // Current service description has profile (L) and assertion (Q)
Receive Monitor Information

1.  while there is a violation
2.       evaluate assertion compatibility
3.       if compatible
4.          return true
5.       else
6.           A(Q) <-- calculate satisfaction value
7.           A_t(Q) <-- satisfaction value using loyalty degree
8.           if A_t(Q) > MinimumValue
9.                Internal Renegotiation // among service instances {Qi}
10.               return true
11.          else
12.               Service Replacement //  in the service composition {Si}
13.          endif
14.      endif
15. endwhile

end algorithm
```

Fig. 2. Pseudo code for adaptation strategies

The algorithm takes one input that is the description of the service defined in the contract. The algorithm starts when it receives information from monitoring engine or when one party requests a new requirement. Assertion compatibility is evaluated in line 2 to identify whether the new change is according to the existing contract based on the mechanism we introduced in [4]. If the change is not compatible it means that it is not within one of the acceptable situations then basically the satisfaction value is calculated (line 6 and 7) to express to what extent the contract is violated (degree of satisfaction). In line 8, the satisfaction value will be compared with a threshold (MinimumValue) that is defined in the contract. According to the degree of satisfaction, an adaptation action will be

chosen to either renegotiate the existing contract with the same service provider (line 9) or perform a service replacement to select a new service provider and set a new contract (line 12). As it is demonstrated in Fig. 1, the internal renegotiation is at the level of assertions and among service instances while service replacement is at the level of profiles between different service types.

7 Adaptation

Having identified the satisfaction value and the compatibility mechanism it is possible to trigger adaptation/evolution strategies. According to the degree of satisfaction we basically propose three different categories, namely Compatible, Soft Adaptation and Hard Adaptation to perform adaptation and evolution as explained in the following:

1. Compatible: This is the situation that the involving parties are considered to be compatible with each other according to the existing contract. For this, we provided in [4] the acceptable conditions in which quality changes can take place without the need for involving parties to change. Therefore, the adaptation action is indeed Do Nothing. This adaptation is applicable for shallow changes and as a result the existing contract will not be modified. The main research challenge in this category is how to provide the compatibility in case the QoS changes.

2. Soft Adaptation: The second adaptation category is considered as an internal renegotiation between the involving parties over the existing contract. This happens when changes are not in one of the acceptable compatible conditions, therefore, the contract is not respected to some extent. According to the degree of satisfaction, this situation could be internally renegotiated so that involving parties could continue with a modified version of the contract (Penalties are applicable in some circumstances). Possible adaptation strategies are relaxing a constraint or goal, reconciliation, compensation, changing value ranges of parameters and so on. Depending on the service description it is possible to use multiple value ranges for the same quality dimension. Therefore, another action would be using an alternative assertion (our approach) or using an OR-refinement of goals in goal oriented approaches [22]. Overall, changes in this category are still considered to be shallow changes however they require a contract modification which is internally and between the involving parties.

3. Hard Adaptation: The third adaptation category takes place when the contract is not valid any longer between the involving parties and there is a high degree of dissatisfaction. Possible strategies in this category include new binding, service selection or re-composition. In these approaches, the cost of service replacement, reconfiguration of the composition and setting a new contract need to be taken into the consideration. Changes are in the deep category of changes and the result of change will propagate in the entire value chain of the service composition.

8 Conclusions and Future Work

This paper concerns the issue of adaptation strategies from the QoS perspective. In SBAs adaptation and evolution of services are necessary due to the various changes, however, there is no overall consensus in selecting the best strategy and the consequences of adaptation are usually neglected. In this paper, we addressed such problematic issues in service adaptation and evolution. In particular, we proposed a flexible approach to deal with QoS parameters formulated within given value ranges in a contract. We applied a fuzzy approach to define quality parameters so that their degree of satisfaction could be estimated. For that purpose we introduced a satisfaction function. Specifically, we propose an algorithm for adaptation strategies that works based on the satisfaction levels.

As part of our future work, we will implement a prototype of the proposed approach. We will also investigate to improve the satisfaction function in order to show a more realistic behavior specially for the Non-fuzzy parameters.

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. Di Nitto, E., Kazhamiakin, R., Mazza, V., Bucchiarone, A., Cappiello, C., Pistore, M.: Design for adaptation of service-based applications: Main issues and requirements. In: The Fifth International Workshop on Engineering Service-Oriented Applications: Supporting Software Service Development Lifecycles, WESOA (2009)
2. Allen, J.F.: Maintaining Knowledge about Temporal Intervals. Communications of the ACM 26(11), 832–843 (1983)
3. Andrieux, A., et al.: Web Services Agreement Specification (WS-Agreement). Recommended standard, Open Grid Forum (March 2007)
4. Andrikopoulos, V., et al.: QoS Contract Formation and Evolution. In: Buccafurri, F., Semeraro, G. (eds.) EC-Web 2010. LNBIP, vol. 61, pp. 290–304. Springer, Heidelberg (2010)
5. Canfora, G., Di Penta, M., Esposito, R., Villani, M.L.: Qos-aware replanning of composite web services. In: ICWS, pp. 121–129 (2005)
6. Cappiello, C., Pernici, B.: Quality-aware design of repairable processes. In: The 13th International Conference on Information Quality (ICIQ 2008), pp. 382–396 (2008)
7. Chafle, G., Dasgupta, K., Kumar, A., Mittal, S., Srivastava, B.: Adaptation inweb service composition and execution. In: IEEE International Conference on Web Services, pp. 549–557 (2006)
8. Harney, J., Doshi, P.: Adaptive web processes using value of changed information. In: Dan, A., Lamersdorf, W. (eds.) ICSOC 2006. LNCS, vol. 4294, pp. 179–190. Springer, Heidelberg (2006)

9. He, Q., Yan, J., Jin, H., Yang, Y.: Adaptation of web service composition based on workflow patterns. In: Bouguettaya, A., Krueger, I., Margaria, T. (eds.) ICSOC 2008. LNCS, vol. 5364, pp. 22–37. Springer, Heidelberg (2008)
10. Keller, A., Ludwig, H.: The wsla framework: Specifying and monitoring service level agreements for web services. J. Network Syst. Manage. 11(1) (2003)
11. Kritikos, K., Plexousakis, D.: Semantic qos-based web service discovery algorithms. In: ECOWS, pp. 181–190 (2007)
12. Li, P., Comerio, M., Maurino, A., De Paoli, F.: Advanced non-functional property evaluation of web services. In: Proceedings of the 2009 Seventh IEEE European Conference on Web Services, ECOWS 2009, Washington, DC, USA, pp. 27–36. IEEE Computer Society, Los Alamitos (2009)
13. Li, P., Comerio, M., Maurino, A., De Paoli, F.: An approach to non-functional property evaluation of web services. In: Proceedings of the 2009 IEEE International Conference on Web Services, ICWS 2009, Washington, DC, USA, pp. 1004–1005. IEEE Computer Society, Los Alamitos (2009)
14. Liu, X.F.: Fuzzy requirements. IEEE Potentials 17(2), 24–26 (1998)
15. Martín-Díaz, O., Ruiz-Cortés, A., Benavides, D., Durán, A., Toro, M.: A quality-aware approach to web services procurement. In: Benatallah, B., Shan, M.-C. (eds.) TES 2003. LNCS, vol. 2819, pp. 42–53. Springer, Heidelberg (2003)
16. Martín-Díaz, O., Ruiz-Cortés, A., García, J.M., Toro, M.: Dealing with fixable and non-fixable properties in service matchmaking. In: Dan, A., Gittler, F., Toumani, F. (eds.) ICSOC/ServiceWave 2009. LNCS, vol. 6275, pp. 228–237. Springer, Heidelberg (2010)
17. De Paoli, F., Palmonari, M., Comerio, M., Maurino, A.: A meta-model for non-functional property descriptions of web services. In: Proceedings of the 2008 IEEE International Conference on Web Services, Washington, DC, USA, pp. 393–400. IEEE Computer Society, Los Alamitos (2008)
18. Paschke, A., Schnappinger-Gerull, E.: A categorization scheme for sla metrics. In: Service Oriented Electronic Commerce, pp. 25–40 (2006)
19. Ran, S.: A framework for discovering web services with desired quality of services attributes. In: ICWS, pp. 208–213 (2003)
20. Sahai, A., Durante, A., Machiraju, V.: Towards automated sla management for web services. Technical Report (2001)
21. Adel Serhani, M., Dssouli, R., Hafid, A., Sahraoui, H.: A qos broker based architecture for efficient web services selection. In: ICWS 2005: Proceedings of the IEEE International Conference on Web Services, Washington, DC, USA, pp. 113–120. IEEE Computer Society, Los Alamitos (2005)
22. Wang, Y., McIlraith, S.A., Yu, Y., Mylopoulos, J.: Monitoring and diagnosing software requirements. Autom. Softw. Eng. 16(1), 3–35 (2009)
23. Zadeh, L.A.: Fuzzy sets. Information and Control 8, 338–353 (1965)
24. 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(5), 311–327 (2004)

CAGE: Customizable Large-Scale SOA Testbeds in the Cloud*

Lukasz Juszczyk[1], Daniel Schall[1], Ralph Mietzner[2], Schahram Dustdar[1], and Frank Leymann[2]

[1] Distributed Systems Group, Vienna University of Technology
lastname@infosys.tuwien.ac.at
[2] Institute of Architecture of Application Systems, University of Stuttgart
lastname@iaas.uni-stuttgart.de

Abstract. Large-scale and complex distributed systems are increasingly implemented as SOAs. These comprise diverse types of components, e.g., Web services, registries, workflow engines, and services buses, that interact with each others to establish composite functionality. The drawback of this trend is that testing of complex SOAs becomes a challenging task. During the development phase, testers must verify the system's correct functionality, but often do not have access to adequate testbeds. In this paper, we present an approach for solving this issue. We combine the *Genesis2* testbed generator, that emulates SOA environments, with *Cafe*, a framework for provisioning of component-based applications in the cloud. Our approach allows to model large-scale service-based testbed infrastructures, to specify their behavior, and to deploy these automatically in the cloud. As a result, testers can emulate required environments on-demand for evaluating SOAs at runtime.

1 Introduction

Service-oriented computing (SOC), based on messages being exchanged among loosely-coupled components, provides a high level of flexibility and scalability which benefits the realization of large-scale and complex distributed systems [1], called service-oriented architectures (SOAs). SOAs do not only comprise Web services, clients, and registries/brokers, as depicted in the Web service triangle [2], but integrate diverse components, such as service buses, message mediators, monitors, governance systems, etc. By applying asynchronous communication via exchanging SOAP messages, and avoiding blocking RPC-like invocations, these systems can potentially scale to large dimensions. Moreover, due to the ability of dynamic binding, SOA systems can integrate new components/services and grow (and shrink) dynamically at runtime. Taking a look at the composition of typical SOAs, two categories of components emerge: (a) stand-alone components, such as single Web services which do not depend on others, and (b) complex components,

* This work is supported by the EU through the projects COMPAS (No. ICT-2008-215175) and S-Cube (No. ICT-2007-215483).

E.M. Maximilien et al. (Eds): ICSOC 2010 Workshops, LNCS 6568, pp. 76–87, 2011.

which interact with others and, this way, manage the SOA and/or establish composite functionality. Obviously, also SOA systems must be tested intensively before final deployment in order to verify their correct execution. For stand-alone components, the testing procedure is well-supported (as outlined in Section 5) or, at least, does not pose new challenges compared to traditional software testing. However, engineers that develop complex components which operate in dynamic SOA environments are facing the problem of how to test their software. They must ensure that their components are able to scale with a growing number of participating services, that quality of service requirements are met, that the software is stable and dependable, etc. Characteristics like these can only be verified by testing the developed component at runtime and in a multitude of real(istic) scenarios. This, however, implies that the component must be deployed in these different designated environments for getting meaningful test results. Unfortunately, testers often do not have access to the designated environments during the development phase, e.g., either because some parts are not available yet or because it integrates commercial external services, which would make testing costly. Furthermore it is often impossible to perform multiple tests (for example, regression tests and load tests) at the same time because the test infrastructure does not offer enough of resources.

In this paper we are trying to solve these issues. We present *Cage*, a framework and methodology for emulating SOA environments (for utilization as testbeds) and for deploying these automatically in the cloud. Our approach combines two complementary frameworks: *Genesis2* [3], for generating SOA testbeds, and *Cafe* [4], for provisioning these across a cloud platform. Cage allows testers to specify *testbed families* consisting of diverse SOA components and variability, to customize their behavior and non-functional properties, and to automatically generate running testbed instances based on the customization. Furthermore, by using the cloud as a platform, Cage provides a convenient and cost-efficient way to set up multiple arbitrarily large testbed instances simultaneously on-demand. In a nutshell, the most distinct contributions of our approach are:

1. Separation of testbed development and testing via *testbed families*
2. A self-service *testbed portal* for testbed customization
3. An infrastructure to provision and run complex, flexible testbeds on-demand

Our approach is presented as follows. Next, we present the motivation for our research. In Section 3 we introduce the two base frameworks Cage consists of. Section 4 describes the combined Cage framework and its functionality. Finally, in Sections 5 and 6 we review related work and conclude our paper.

2 Motivation: Large-Scale SOA Testbed Infrastructures

Loose coupling and dynamic binding have always been listed among the most important features of Web service-based SOA. By avoiding static interactions but being able to choose services dynamically, SOAs are capable of incorporating new participating services at run-time and to cope with unavailable ones

by rebinding to alternatives. This flexibility makes it possible to build systems which are not restricted to an environment of fixed size and topology, but are able to deal with dynamic and large-scale ones. Let us take Amazon Mechanical Turk[1] (MTurk), a crowdsourcing platform, for example. MTurk registers Web services of human workers and provides these to clients which incorporate the offered functionality into their applications/workflows. MTurk must be able to handle load peaks, scale with the number of registered services and consuming clients, and, despite all possible difficulties, provide stable and dependable services. However, loose coupling, dynamic binding, or other typical SOA-features do not solve the scalability issue per se. The system's internal mechanisms must be still able to cope with a high number of partner components (e.g., services, clients, workflow engines) and incoming requests. During the development of such systems the question appears of how to test these mechanisms and how to verify their correct execution in critical scenarios. These scenarios can comprise thousands of components which have diverse functional as well as non-functional properties, and, therefore, put high load on the system. Setting up such scenarios for testing purposes is an intricate task. In a nutshell, testers are confronted with the problem of a) how to create testbeds which emulate realistically the final deployment environment and b) where to host large-scale testbeds in a cost-efficient manner.

We have elaborated on the first issue in [3] by showing how our Genesis2 framework supports the generation of customizable SOA testbed infrastructures. In our approach testers model SOA environments and Genesis2 takes care of generating and deploying running testbed instances implementing the modeled behavior. Of course, an adequate back-end hardware infrastructure is required in order to be able to host all generated SOA components, which can get very costly for large-scale testbeds. In the last years cloud computing emerged as an interesting solution to this problem, as it enables users to rent hardware on-demand, also referred to as Infrastructure as a Service (IaaS). Instead of buying expensive hardware for hosting testbeds, which is most likely running idle in periods when no test runs are performed, engineers can rent remote hardware for an arbitrary time and quit the service when it is no longer required. In our Cage approach we make intense use of IaaS. We apply the Cafe framework which supports automatic allocation hosts/servers in the cloud and, this way, provides a flexible hosting infrastructure for SOA testbeds. The most significant contribution of our approach is that we enable testers to generate functional SOA testbeds on-demand on a dynamically allocated back-end infrastructure.

3 Base Frameworks for CAGE

Cage derives it functionality (and name) from combining two base frameworks: *Cafe* and *Genesis2*. Cafe provides support for an automated provision of component-based applications into the cloud, while Genesis2 makes use of the infrastructure provided by Cafe and generates customizable and dynamic SOA

[1] http://www.mturk.com/

testbeds. In the following these frameworks are shortly introduced to outline their concepts.

3.1 Cafe

Cafe (composite application framework) [4] is a framework for definition, customization and automatic provisioning of complex composite applications in the cloud. Cafe is centered around the basic concept of an *application template* which consists of an *application model* and a *variability model*. The application model describes the basic components of the application, whereas the variability model describes variability of the application. Application templates are offered to customers by a provider in an *application portal*. The customers are guided through the customization of the template by a *customization flow* that is generated from the variability model of the application. Once a the template is transformed into an customer-specific *application solution*, this solution is then automatically provisioned by the *provisioning infrastructure*. Cafe makes use of the interfaces of different cloud providers, such as Amazon EC2[2], to setup components.

A Cafe application model contains a set of components that realize the functionality of the application. Components can be arbitrary elements of an application such as middleware components that are supplied by a provider, or components where the code is shipped with the application (*internal components*), such as services, Web applications, or business processes. Provider-supplied components must have a special *component type* that indicates a class of components such as JBOSS application server components. Internal components are of a certain *implementation type*, e.g., JEE application or BPEL process. Component types define if components of a certain implementation type can be deployed on them. Components can have deployment relations among them, indicating that one component must be deployed on another component. Application template developers use internal and provider supplied components and their deployment relations to describe their application.

3.2 Genesis2

The purpose of the Genesis2 framework [3] (in short, G2) is to support the setup of testbeds for SOA. It allows to emulate environments consisting of services, clients, registries, and other SOA components and to program their behavior. G2's most distinct feature is its ability to generate running testbed instances (in contrast to performing pure simulations) and to integrate these into existing SOA environments, which empowers testers to evaluate SOA systems at runtime.

G2 comprises a centralized front-end, from where testbeds are modeled and controlled, and a distributed back-end, consisting of hosts (in Cage these are located in the cloud) at which the models are transformed into real testbed instances. The front-end maintains a virtual view on the testbed, allows to manipulate it on-the-fly via scripts, and propagates changes to the back-end in order

[2] http://aws.amazon.com/ec2/

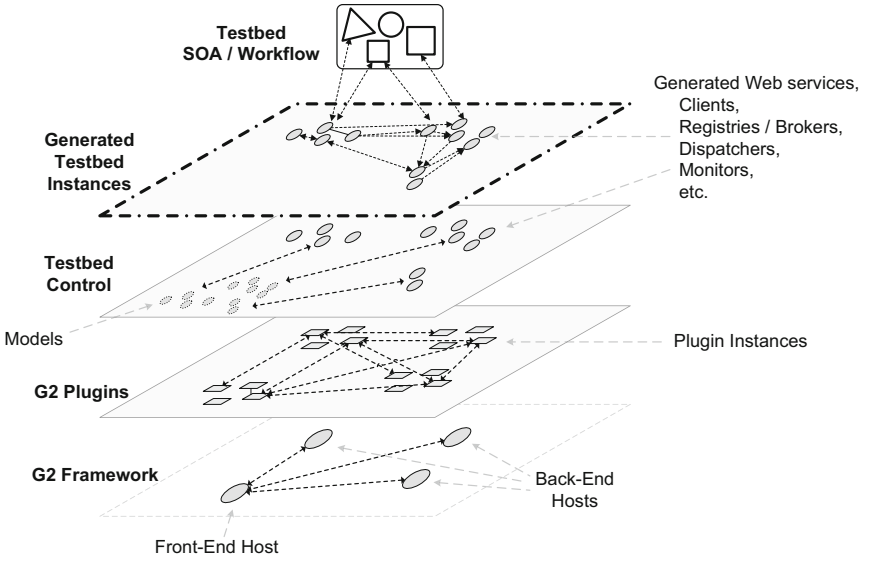


Fig. 1. Layered topology of a G2 testbed

to adapt the running testbed. For the sake of extensibility, G2 uses composable plugins which augment the testbed's functionality, making it possible to emulate diverse topologies, functional and non-functional properties, and behavior. Fig. 1 depicts a simplified view on the different layers of a G2-based testbed and the interactions within them. At the two bottom layers, G2 connects the front-end to the distributed back-end and installed plugins establish their communication structures. Most important are the two top layers. Based on the provided model schema, the tester creates models of SOA components which are then being generated and deployed at the back-end hosts. At the very top layer, the testbed instances are running and behave/interact according to the specification. The aggregation of these instances constitutes the actual testbed infrastructure on which the developed SOA can be evaluated.

In summary, at the front-end the tester specifies via Groovy scripts the testbed details, defining *what* shall be generated *where*, with *which customizations*, and the framework takes care of synchronizing the model with the corresponding back-end hosts on which the testbed elements are generated and deployed. The following snippet contains a sample script for modeling a Web service, programming it's behavior (in this case just returning a String value), and deploying it at a back-end host. After deployment, it is possible to perform adaptations on-the-fly by changing the Web service's model which is immediately propagated to the generated instance at the back-end.

```
// import reference of cloud back-end host
def beHost1 = host.create("someHost:8080")

// import data type from XSD file
def dt = datatype.create("/path/to/types.xsd","vCard")
```

```
// create model of TestService with one operation
def service = webservice.build {

    TestService(binding: "doc,lit") {
       SayHi(card: vCard, result: String) {
          return "hi ${card.name}"
       }
    }
}[0]

service.deployAt(beHost1) // deployment

service.operations+= ... // on-the-fly adaptation of deployed service
```

4 The Cage Framework

4.1 Cage Methodology and Roles

The Cage methodology specifies how the framework can be used to generate large-scale customizable SOA testbeds and to deploy them on-demand in cloud-based infrastructures. The methodology comprises three steps (*modeling, setup, & execution*) that are performed by two different roles (*testbed engineer & tester*) during the establishment of Cage testbeds. Fig. 2 depicts a high-level overview of the three steps and the Cage tools supporting them.

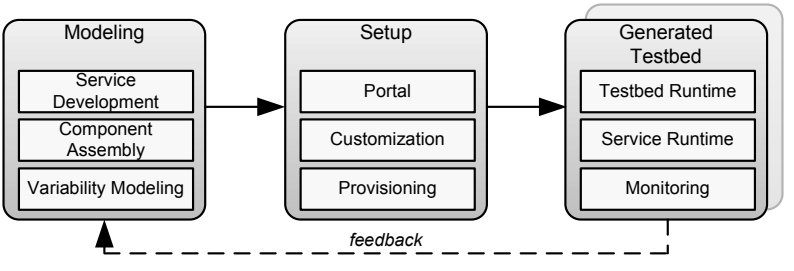


Fig. 2. Overall Cage approach and architecture

Modeling. At first, in the modeling step, the testbed engineer creates a specification/model of the testbed. The model defines the composition and topology of the testbed's components and, in addition, allows the testbed engineer to program the functional behavior. Due to the extensible nature of Genesis2, it is possible to model diverse types of testbed components, such as Web services, clients, registries, and message dispatchers, and to compose these into an emulated SOA consisting of mock-up components as well as optional own and third-party services integrated into the testbed. To be able to automatically deploy the whole testbed infrastructure, these components and their deployment relations are modeled in the component assembly modeling tool in which the testbed engineer defines the variability for the testbed, for example, different qualities of services and/or functional aspects.

Setup. Based on the created model, the testbed engineer uploads the corresponding artifacts to a portal. Testers have then the ability to request instances of the uploaded model via a control interface (part of the portal). A testbed model can be customized according to different aspects and requirements. For example, customization could be based on different perturbation and fault handling strategies, as show in [5]. Customization is accomplished by binding points of variability, e.g., by selecting one of multiple alternatives or by entering values for a point of variability.

Provisioning. The final step is to provision the testbed including the test services (emulated behavior), real services and middleware components. This is done automatically by the Cage framework once all variability has been bound by the tester.

The monitoring layer of the generated testbed captures various activities within the testbed environment such as service invocations and lookup requests (registry access). Low level logs are aggregated into metrics to analyze statistical variation of service behavior, as explained in [6]. The tester has the ability to analyze these metrics (using visualization tools) and to adjust models of variability accordingly. This cycle is depicted through the feedback arrow in Fig. 2.

4.2 Modeling Testbeds

Fig. 3 shows a component assembly example to illustrate various Cage concepts. Component assembly in Cage closely follows the Cafe approach [4] where modeling of composite application templates in Cafe is centered around components. Cage introduces a new component type to Cafe, namely the *Genesis Framework* component type. It represents a middleware on which certain other components can be deployed, namely those that have an implementation type of *Genesis Test Service*. This notion is similar to other component types in Cafe, such as the component types JBoss and Apache ODE shown in Fig. 3 which allow to deploy components of implementation type JEE Application and BPEL Process on them, respectively.

As a result, the testbed engineer can compose a testbed by combining middleware components including the Genesis Framework, application servers, Web servers and process engines. Also, components can be composed such as test services, real services, Web applications or business processes that run on the aforementioned middleware components. This enables engineers to systematically define complex testbed scenarios with the support of the Cage approach.

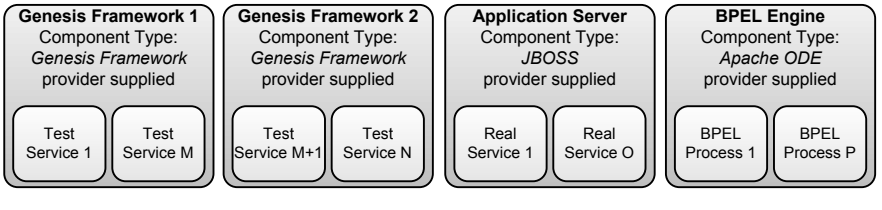


Fig. 3. Component assembly example

In Cafe, application templates are annotated with a *variability model* that is specified using the Cafe *variability meta-model* [4]. Such a variability model contains a set of *variability points* that specify possible configuration *alternatives* (see Fig. 4). Different types of alternatives exist that can be arbitrarily combined in one variability point:

- *Explicit alternatives* allow to specify a concrete value that can be selected, i.e. a fixed value for a delay.
- *Expression alternatives* allow to specify an expression (in XPath) that is evaluated and calculates the value that is entered at the variability point, for example, based on the values entered at another variability point.
- *Free alternatives* allow to prompt a user for an input, for example, to specify a specific failure rate.

Each variability point contains one or more *locators* that point into documents of the template that the variability point affects. Variability points can have complex *dependencies* that indicate that one or more variability points depend on one or more other variability points. These dependencies allow to specify temporal dependencies, i.e., a variability point can only be bound after all variability points that it depends on are bound. *Enabling conditions* can be defined for each variability point that specify under which condition which alternatives of this variability point can be chosen. This allows to specify complex dependencies such as "if the response-time of component A is greater than 2s the response-time of component B must be greater than 3s". In a Cage testbed a variability point can, for example point, into the WSDL document of a BPEL process to customize the endpoint of a test-service that should be invoked by that process. Thus, a variability point can indicate provisioning time that must be filled by the provisioning environment. Variability points can also express functional and

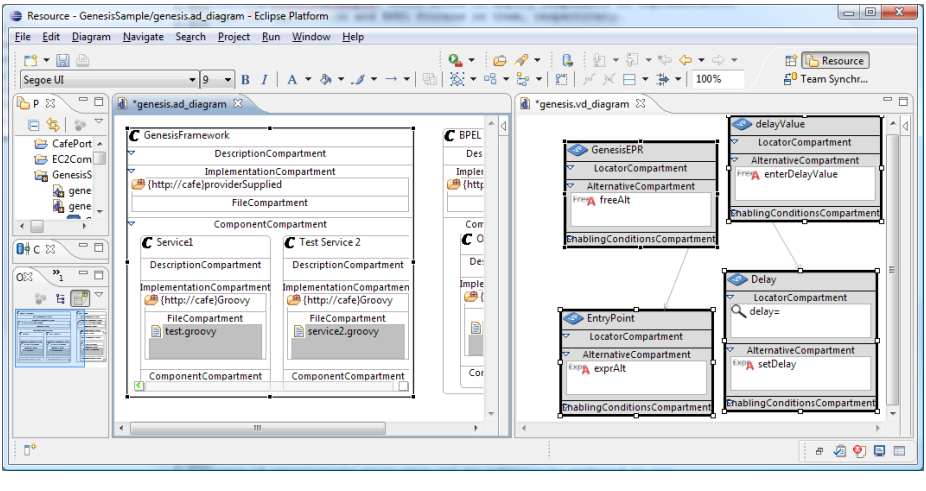


Fig. 4. Cage modeler: testbed components on the left, variability model on the right

non-functional variability. For example, a locator of a variability point can point into a Groovy script implementing a test service to configure its behavior or the average response-time this service should simulate.

4.3 Testbed Setup

Once a testbed, including its variability, has been modeled by the testbed engineer, it can be offered to testers for retrieval via the *test portal*. The tester is guided through the setup process via a *customization flow* which is a workflow generated from the variability model of a testbed, as introduced in [7]. The customization flow prompts the tester for all necessary decisions. In addition to the configuration of the testbed the tester specifies for how long the testbed is to be used. The duration is an important property as it allows to free the used resources for testing after a pre-defined amount of time. Once a tester has customized the testbed for the particular test to be performed (for example, the testbed is configured for load-testing instead of regression testing), the provisioning infrastructure sets up the necessary components and configures them as described in the next section.

4.4 Testbed Provisioning

After a testbed has been customized, it is being generated, deployed, and provided to the tester. This procedure comprises these steps:

- *Component provision & deployment*: Components available in the infrastructure are bootstrapped via their corresponding *provisioning services* [8]. For example Genesis2 back-end instances, workflow engines, and other base components are started, in order to deploy test services, real services and test clients on top.
- *Component configuration*: Components are configured by the provisioning infrastructure in order to establish links among then and to create a composite testbed. For example, a BPEL process that orchestrates a set of test-services must be configured with the endpoints of these test services as specified in the variability model for the respective testbed.

Testbed provisioning is done via a *provisioning flow* that is generated by the Cage framework from the model of the testbed. The provisioning flow respects the component dependencies introduced by the deployment relations (i.e. which component must be deployed on which other component) and the variability dependencies (i.e. which component must be configured with properties of which other component).

The following code snippet shows parts of a simplified G2 testbed template script which contains placeholders (uppercase strings) to be customized by Cafe at deployment time. Moreover, it returns a list of URLs of the deployed service endpoints, to be passed to other components. This provides a convenient method for the parameterized deployment of cloud-based testbeds.

```
// placeholder for references to cloud back-end host
def hostList = {{BACKENDHOSTS}}

def rand=new java.util.Random()
def randomHost = { -> hostList[rand.nextInt(hostList.size())] }
urlList=[]

serviceList.each { s->
    s.qos.responsetime={{QOS_DEF_RESPONSETIME}}
    s.qos.availability={{QOS_DEF_AVAILABILITY}}

    h=randomHost()
    urlList+=s.deployAt(h) // deployment at random host, collect URLs
}

return urlList
```

5 Related Work

Today, testing of complex service-based applications is a tedious task. Setting up of complex testbeds requires the acquisition and setup of computing, middleware and software resources which in most enterprises takes a considerable amount of time and effort to do.

To overcome this burden, several authors propose to use cloud resources that can be acquired on-demand, to deploy complex testbeds [9,10]. However, deploying the components to be tested on these cloud resources is still manual labor in these approaches and thus is time-consuming and error-prone. To automate the setup of complex component based applications in a reproducible way, automated provisioning environments [4,8,11] have been proposed. These environments first require the modeling of *component topologies* including the required components and their relations among each other. In this paper we investigate how this modeling affects the testing of complex service-based applications. In particular the modeling of the testbed components and their variability is similar to the *domain engineering phase* in software product line engineering in which a platform is developed that can then be customized into runnable applications in the *application engineering phase* [12,13]. Thus we adapt the notion of *testbed platform* as the basic testbed artifacts that can be customized into a concrete testbed. The application engineering phase in software product line engineering then corresponds to the customization and deployment of a testbed into a running testbed instance.

Several approaches have been developed which could be applied for testing adaptation mechanisms. SOABench [14] and PUPPET [15], for instance, support the creation of mock-up services in order to test workflows. However, these prototypes are restricted to emulating non-functional properties (QoS) and cannot be enhanced with programmable behavior. By using Genesis2 [3] which allows to extend testbeds with plugins we were able to implement a testbed which was flexible enough to test diverse adaptation mechanisms.

6 Conclusion and Future Work

In this paper we introduced the Cage framework to model, setup and automatically provision large-scale SOA testbeds in the cloud. We introduced the role of testbed engineers that model testbed families including their variability. Such a testbed family is then made available in a testbed portal for testers. Instead of manually configuring and deploying a large-scale SOA testbed, a test engineer can select the required testbed family from the portal, customize it using a generated customization wizard, and have it automatically deployed by the Cage infrastructure. Compared to existing approaches, the Cage approach facilitates testing, saves setup and configuration time, and enables testing in the cloud taking full advantage of pay-per-use models of, for example, IaaS providers.

In future work we will investigate how the results of individual test-runs can be used to automatically derive certain parameters of a testbed in order to iteratively optimize a given system to avoid cumbersome (low-level) configuration of the corresponding production system. Also, we plan to further implement metrics for testbed analytics and tools for visualizing complex behavior in cloud-based testbeds.

References

1. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F.: Service-oriented computing: a research roadmap. Int. J. Cooperative Inf. Syst. 17(2), 223–255 (2008)
2. Huhns, M.N., Singh, M.P.: Service-oriented computing: Key concepts and principles. IEEE Internet Computing 9(1), 75–81 (2005)
3. Juszczyk, L., Dustdar, S.: Script-based generation of dynamic testbeds for soa. In: ICWS 2010, pp. 195–202. IEEE Computer Society, Los Alamitos (2010)
4. Mietzner, R., Unger, T., Leymann, F.: Cafe: A generic configurable customizable composite cloud application framework. In: Meersman, R., Dillon, T., Herrero, P. (eds.) OTM 2009. LNCS, vol. 5870, pp. 357–364. Springer, Heidelberg (2009)
5. Juszczyk, L., Dustdar, S.: Programmable Fault Injection Testbeds for Complex SOA. In: Maglio, P.P., Weske, M., Yang, J., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6470, pp. 411–425. Springer, Heidelberg (2010)
6. Psaier, H., Juszczyk, L., Skopik, F., Schall, D., Dustdar, S.: Runtime Behavior Monitoring and Self-Adaptation in Service-Oriented Systems. In: SASO, pp. 164–174. IEEE, Los Alamitos (2010)
7. Mietzner, R., Leymann, F.: Generation of bpel customization processes for saas applications from variability descriptors. In: IEEE SCC (2), pp. 359–366. IEEE Computer Society, Los Alamitos (2008)
8. Mietzner, R., Leymann, F.: Towards provisioning the cloud: On the usage of multi-granularity flows and services to realize a unified provisioning infrastructure for saas applications. In: SERVICES I, pp. 3–10. IEEE Computer Society, Los Alamitos (2008)
9. Varia, J.: Cloud architectures. Amazon white paper (2008)
10. Ciortea, L., Zamfir, C., Bucur, S., Chipounov, V., Candea, G.: Cloud9: a software testing service. SIGOPS Oper. Syst. Rev. 43(4), 5–10 (2010)
11. Arnold, W., Eilam, T., Kalantar, M.H., Konstantinou, A.V., Totok, A.: Pattern based soa deployment. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749, pp. 1–12. Springer, Heidelberg (2007)

12. Bosch, J.: Design and use of software architectures: adopting and evolving a product-line approach. Addison-Wesley Professional, Reading (2000)
13. Pohl, K., Böckle, G., Van Der Linden, F.: Software product line engineering: foundations, principles, and techniques. Springer, Heidelberg (2005)
14. Bianculli, D., Binder, W., Drago, M.L.: Automated performance assessment for service-oriented middleware. Technical Report 2009/07, Faculty of Informatics - University of Lugano (November 2009)
15. 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)

Engineering High Performance Service-Oriented Pipeline Applications with MeDICi

Ian Gorton, Adam Wynne, and Yan Liu

Pacific Nothwest National Lab
PO Box 999, Richland WA 99352, USA
{adam.wynne,yan.liu,ian.gorton}@pnl.gov

Abstract. The pipeline software architecture pattern is commonly used in many application domains to structure a software system. A pipeline comprises a sequence of processing steps that progressively transform data to some desired outputs. As pipeline-based systems are required to handle increasingly large volumes of data and provide high throughput services, simple scripting-based technologies that have traditionally been used for constructing pipelines do not scale. In this paper we describe the MeDICI Integration Framework (MIF), which is specifically designed for building flexible, efficient and scalable pipelines that exploit distributed services as elements of the pipeline. We explain the core runtime and development infrastructures that MIF provides, and demonstrate how MIF has been used in two complex applications to improve performance and modifiability.

Keywords: middleware, software pipelines, SOA, component-based systems.

1 Introduction

In many application domains in science and engineering, data produced by sensors, instruments and networks is commonly processed by software applications structured as a pipeline [1]. Pipelines comprise a sequence of software components that progressively process discrete units of data to produce a desired outcome. For example, in geo-sciences, data on geology extracted from underground well drilling can be processed by a simple software pipeline that converts the raw data format to XML, extracts information from the files about drilling locations and depths, displays the location on a map-based interface and creates an entry in the database containing metadata for searching.

In many applications, simple linear software pipelines are sufficient. However, more complex applications require topologies that contain forks and joins, creating pipelines comprising branches where parallel execution is desirable. It is also increasingly common for pipelines to process very large files or high volume data streams which impose end-to-end performance constraints. Additionally, processes in a pipeline may have specific execution requirements and hence need to be distributed as servicesacross a heterogeneous computing and data management infrastructure.

From a software engineering perspective, these more complex pipelines become problematic to implement. While simple linear pipelines can be built using minimal

E.M. Maximilien et al. (Eds): ICSOC 2010 Workshops, LNCS 6568, pp. 88–99, 2011.

infrastructure such as scripting languages [2], complex topologies and large, high volume data processing requires suitable abstractions, run-time infrastructures and development tools to construct pipelines with the desired qualities-of-service and flexibility to evolve to handle new requirements.

In this paper we describe our MeDICi Integration Framework (MIF) that is designed for creating high-performance, scalable and modifiable software pipelines. MIF exploits a low friction, robust, open source middleware platform and extends it with component and service-based programmatic interfaces that make implementing complex pipelines simple. The MIF run-time automatically handles queues between pipeline elements in order to handle request bursts, and automatically executes multiple instances of pipeline elements to increase pipeline throughput. Distributed pipeline elements are supported using a range of configurable communications protocols, and the MIF interfaces provide efficient mechanisms for moving data directly between two distributed pipeline elements.

MIF has been used at the Pacific Northwest National Laboratory (PNNL)in diverse application domains such as bioinformatics [3], scientific data management, cybersecurity, power grid simulation [4] and climate data processing [5]. In this paper, we briefly present the basic elements of MIF for building pipelines, and describe how these mechanisms can become the core of a SOA-based pipeline solution. We then describe the MIF Component Builder that provides a MIF design tool, and finally describe two case studies that demonstrate MIF's capabilities for text processing and cybersecurity.

2 Related Work

Various approaches exist for constructing software pipelines. The most common is to using scripting languages such as Perl, Python or shell scripts [7]. These work well for simple pipelines, as the lightweight infrastructure and familiar programming tools provide an effective development environment. Recently this approach has been extended in the Swift project [9], which has created a custom scripting language targeted at applications running on grid and high performance computing platforms. Swift has a number of attractive features for describing pipelines, but its implementation is limited in scope to large computational infrastructures which provide some necessary runtime infrastructure, for example, job scheduling.

Many workflow tools are also perfectly adequate for creating pipelines. Tools such as Kepler, Taverna and implementations of BPEL [6] can be used to visually constructapipeline-style workflow, link in existing components and services through a variety of protocols, including Web services, and then deploy and orchestrate the pipeline. These tools and approaches work well, but are heavyweight in terms of development infrastructure (ie they require visual development and custom workflow languages) and runtime overheads. The runtime overheads especially make them inappropriate for building pipelines that need to process high volume data streams, and small message payload sizes where fast context switching, lightweight concurrency and buffering are paramount.

Custom approaches also exist for building pipelines. Prominent in this category is Pipeline Pilot [1]. It provides a client-server architecture similar to that of most

BPEL-based tools. The client tools are used to visually create pipelines, in a custom graphical language, based upon underlying service-oriented components. The server executes pipelines defined by the client, orchestrating externally defined components through SOAP-based communications. The client also provides a proprietary scripting language to enable custom manipulation of data as it flows between the main steps of the pipeline.

The MeDICi Integration Framework (MIF) attempts to overcome the collective limitations of the above approaches by:

- allowing construction of pipelines using a standard, simple Java API
- providing a relatively lightweight runtime infrastructure (compared with other Java-based pipeline and workflow creation tools) suitable for both large data handling and long running computations, as well as bursty stream-based data with rapid, lightweight processing needs
- providing a strong separation of concerns between component behavior, inter-component communications and pipeline topology

3 MeDICi Integration Framework Overview

The MIF middleware platform is designed for building applications based on processing pipelines that integrate various software components using an asynchronous messaging platform. Figure 1 below shows a simple example of a processing pipeline with two analysis components.

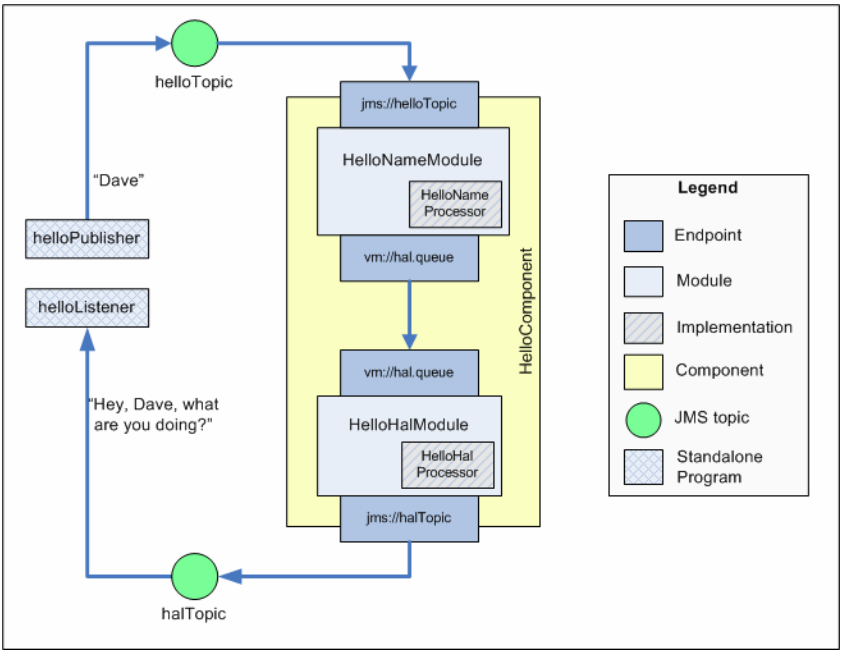


Fig. 1. A "Hello World" Pipeline in MIF

The key concepts in MIF API are:

- **MIF pipeline:** A processing pipeline is a collection of components connected as a directed graph. Data arrives at the first component in the pipeline, and the output of this component is passed to one or more downstream components for further processing. A pipeline can be of arbitrary length, components can accept inputs from one or more upstream components, and send their results to one or more downstream components.
- **MIF Module:** A MIF module is a user-supplied piece of code that inputs some defined data, processes it, and outputs some results. This code is wrapped by the MIF API to make it possible to plug the module in to a processing pipeline.
- **MIF Components:** MIF components provide a way to hierarchically compose MIF modules. A MIF component encapsulates a mini-pipeline of MIF modules.
- **Endpoints:** MIF modules define inbound and outbound endpoints as their connections to other modules in a pipeline. To connect two modules, the outbound endpoint from one module is configured to communicate with an inbound endpoint of another module.
- **MIF Container:** MIF components execute in the MIF container within a Java Virtual Machine. The container can automatically invoke multiple instantiations of pipeline component to handle multiple concurrent messages, and provides asynchronous, buffered communications mechanism to smooth out differences in the processing times of communicating components. This gives MIF pipelines the ability to seamlessly scale on multi-core platforms simply by adding more processing nodes and memory.
- **Local Components:** Local components are Java codes that are constructed using the MIF API. Local components execute within the MIF container.
- **Remote Components:** Remote components execute outside the MIF container, either on the same machine or on another node in a distributed application. The component code can be Java (e.g. an Message-Driven Bean), a Web service, HTTP server, a socket server or an executable written in any programming language. The MIF remote component API wraps the code and provides the capabilities to communicate with it from a MIF pipeline using a configurable protocol.
- **Messages:** Components pass messages down the pipeline as the incoming data is progressively transformed. Complete messages can be simply copied between components, or a reference to the message payload as a URIcan be exchanged to reduce the overheads of passing large messages.

Below is a simple example of configuring the MIF pipeline depicted in Figure 1.

```
// create the pipeline
MifPipeline pipeline = new MifPipeline();

// specify the JMS Connector
pipeline.addMifJmsConnector("tcp://localhost:61616",
JmsProvider.ACTIVEMQ);

//Add a component to the pipeline
HelloWorldComponent hello = new HelloWorldComponent();
pipeline.addMifComponent(hello);
```

```
// connect the component's endpoints and start processing
hello.setInNameEndp("jms://topic:NameTopic")
hello.setOutHalEndp("jms://topic:HalTopic")
pipeline.start();
```

The MIF container environment is provided by Mule[1], an open source messaging platform. For this reason, MIF shares many fundamental concepts with widely-used Enterprise Application Integration (EAI) and SOA-based integration brokers. Importantly for our implementation, Mule is lightweight, robust and scalable, traits which MIF inherits.

MIF extends the Mule API to make component-based pipeline construction simpler and to create an encapsulation mechanism for component creation. Our current implementation uses the ActiveMQ JMS for reliable message exchange, but the MIF API is designed as agnostic to the specific messaging platform. This allows deployments to configure MIF applications using JMS providers that meet their quality of service requirements.

4 MIF Component Builder

In order to simplify the creation of MIF pipelines, we created the MIF Component Builder to enable programmers to visually create and test pipelines inside of Eclipse Integrated Development Environment (IDE). Using this tool, programmers can visually create a pipeline, generate stub code for *MifModules* and *UserImplemented* entities, progressively implement the necessary code stubs, and run the pipeline inside the IDE. Thisprovides round trip development in which the programmer can cycle through design, generate and test until the pipelineis ready to be deployed.

We leveraged Model-driven Development (MDD) techniques in the Component Builder that enable the automated creation of model editors. Model-driven software development formalizes abstract software models as the basis for automated transformation into other models and/or programming code. This formalization allows for automated tools to operate on one or more models to generate applications. At the heart of these tools is the Eclipse Modeling Framework[2] (EMF). In addition to a modeling framework, EMF provides a code generation facility and runtime environment that allows models to be expressed in multiple formats and converted to a standard XML interchange (XMI) based format.

The MIF metamodel (a portion of which is shown in Figure2) is specified in EMF's ecore format. In the lower left of Figure2 is the *Pipeline*class, which is the root object of any MIF program. A MIF *Pipeline*comprises one or more *MifComponents*, which are high-level groupings of processing elements known as *modules*. *MifComponent* can be thought of as a sub-pipeline that contains one or more modules which all inherit from *BaseModule*. Modules are linked to each other using communication endpoints: *InboundEndpoint* allows data to be received by a module and *OutboundEndpoint* allows modules to send data to other modules. The metamodel is populated using the Graphical Modeling Framework (GMF), which provides

[1] http://www.mulesoft.org/
[2] http://www.eclipse.org/modeling/emf/

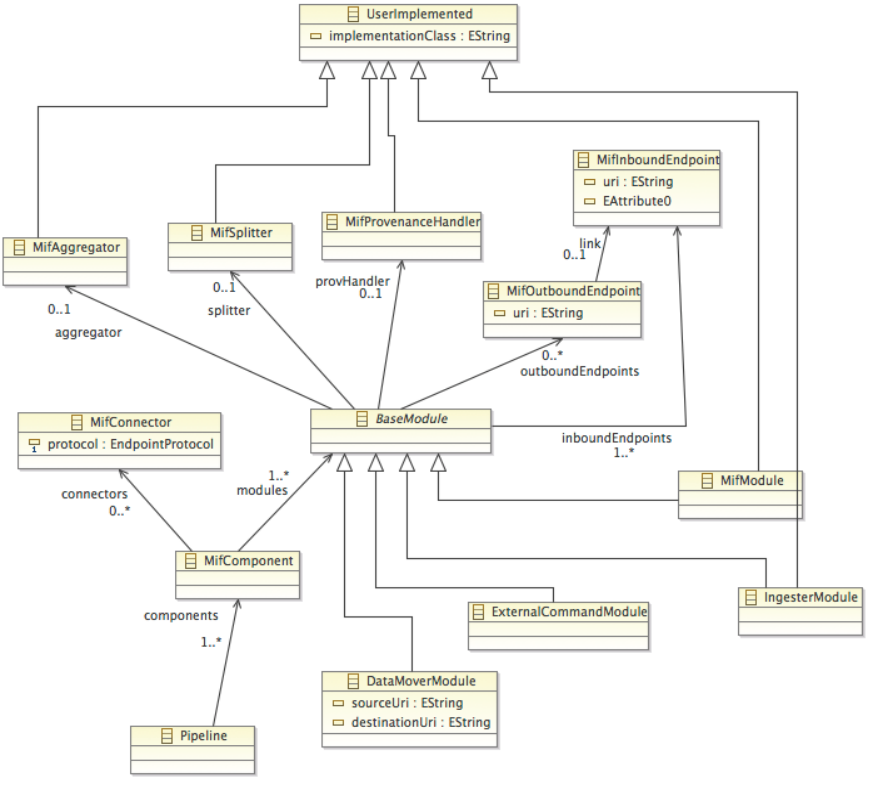


Fig. 2. MIF Base Component Model

model-driven tools for designing and generating graphical user interfaces. Once the metamodel is populated with the design of a particular MIF application, code generators produce Java that calls the MIF API. EMF therefore forms the basis for other tools that add capabilities for graphical editing, advanced template-based code generation, and model-to-model transformations, all of which are used to construct the MIF Component Builder.

5 Case Studies

5.1 Semantic Text Processing Pipeline

Performing semantic analysis of textual documents is a computationally intensive problem that is often structured as a pipeline. The first stage analyzes language structure, breaking down paragraphs and sentences into their constituent grammatical parts, and passes a marked-up version of the document to the next stage. Subsequent steps may for example identify places, events and people. In our application, the final stage compares the semantically marked up documents with an ontology and stores the relevant elements in a datastore as RDF triples.

The application was originally designed using the Apache UIMAtechnology[3]. The key UIMA component is an*annotator* that encapsulates text processing logic in a Java class. Multiple annotators providing specific text processing functions were built into a pipeline to handle each document. Each annotator produced a Java objectcontaining therelevant markups for processing by the downstream annotators. The output from one annotator is subsequently fed into the next annotator to discover more complex relations. Annotators are complex but can be designed independently and configured as needed for the specific application purposes.

The original system was developed as a monolithic UIMA application, as shown in Figure 3. The UIMA analysis engine (AE) is a container that hostsa pipeline of annotators. The configuration of these annotators is described by an XML file that specifies the detailed topology of the annotators, and inputs/outputs of each annotator. The UIMA AE loads the XML file and executes the pipeline of annotators.

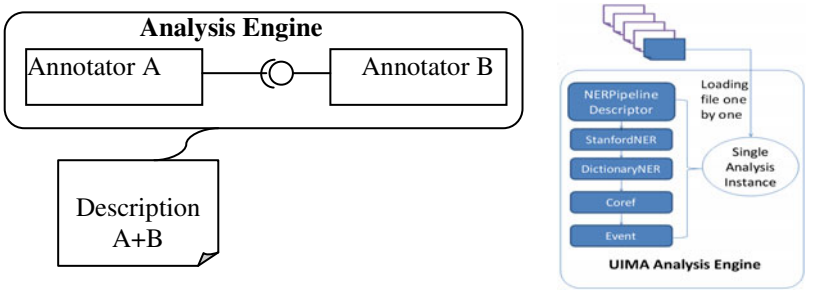


Fig. 3. Architecture of the UIMA pipeline

The monolithic UIMA architecture is straightforward to deploy on a single node, but UIMA has no native mechanism to expose the pipeline as a service.Further, it has several limitations in terms of performance and scaling. First, the pipeline is single-threaded, processing a single document at a time to completion. Hence parallel processing of documents requires multiple instances of the UIMA container to run concurrently. In addition, the processing time of each annotator on a single file was diverse. Two of the 5 annotators in the pipeline consumed 98% of the runtime. As a result, this tightly coupled architecture wascumbersome and heavyweight to scale to process tens of thousands of documents, especially as asingle document can take hundreds of seconds to emerge from the pipeline.

Hence an architectural solution was essentialto decouple the annotators so that optimization strategies such as introducing concurrency to individual annotators could be achieved.This required refactoring the original pipeline into separate annotation services that could be connected through MIF endpoints, as shown in Figure 4. AEs are deployed one separate nodes and host one or more annotation services. For

[3] `http://uima.apache.org/`

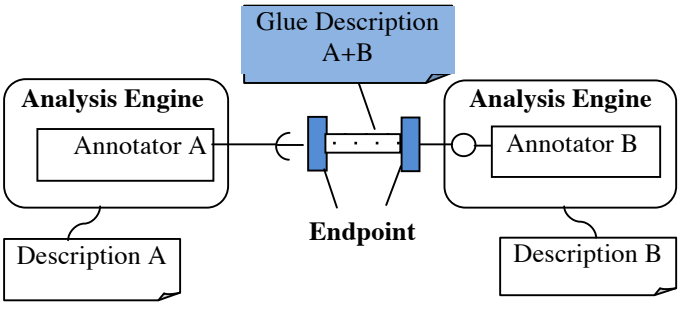


Fig. 4. Service Oriented Annotators Architecture

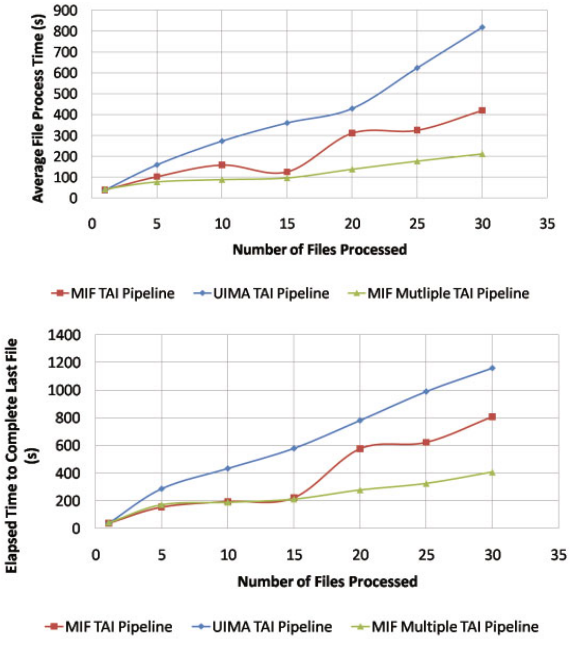


Fig. 5. Performance of threealternative architectures

example, the most time consuming annotator is deployed on one AE, while several other annotators demanding less processing time are bundled into another AE.

The challenge of implementing such an SOA is to compose annotation services into a pipeline that preserves the original analysis properties. We achieve this using MIF, which provides the essential mechanisms to glue together the distributed annotation services. The MIF solution wrapsanAE as a MIF component, and configures the components into a pipeline, which is exposed as a Web service.

In our solution, we devised two architecture alternatives that implement different refactoring strategies using MIF. The alternatives were:

1. A single MIF pipeline containing a MIF file ingester and MIF components wrapping individual annotators. The concurrency required for processing multiple documents simultaneously in the pipeline is managed by MIF by default. As a result, concurrent documents are efficiently processed by the MIF pipeline.
2. Three MIF pipelines are constructed - two for annotators dominating the document processing times, and one for the remainder. This creates a partitioned pipeline architecture. Each MIF pipeline is responsible for the internal concurrency of its MIF components, and the pipelines are connected by JMS endpoints.

We deployed the original architecture and the two refactored architectures using MIF on the same computing environment. We measured individualdocument processing times and the elapsed time to complete the last file in a batch. Figure 5shows the performance as the load of documents increases.

The results demonstrate that the refactored MIF pipelines improve performance significantly. The multiple MIF pipeline architecture scales linearly as the load increases and produces stable performance compared to the single MIF pipeline architecture. The reason is that multiple MIF pipelines balance the annotator workload better through more efficient resource utilization, giving significant performance improvements of up to a factor of 4.

5.2 Cyber Analytic Pipeline

The analysis of network data has traditionally been done in a forensic fashion in which firewall, host, and router logs(referred to as network flows) are stored for post-mortem searching in response to a suspected attack.Purpose built scripts, command-line tools, and graphical tools are manually used to comb through massive amounts of log activity. The workflow of this activity is a manual pipeline in which a set of files are retrieved and transformed from different sources into a common format. Next, programs are used to filter out data that represents normal traffic and find data items that resemble a certain traffic pattern or contain a set of IP addresses of interest.

This workflow presents several challenges, including: (1) repeating successful pipeline runs is challenging due to the number of ad hoc steps taken, (2) swapping out different transformation, analytics, and visualization codes is labor intensive and error prone, and (3) it does not scale to the network security problems encountered today in which data from entire networks must be rapidly aggregated and processed. Automated Intrusion Detection Systems (IDS) such as Snort[4]exist to alleviate some of the manual analysis involved in this pipeline. However, these are typically signature based and suffer from known problems of a high rate of false positives and the inability to detect attacks that are not known in advance.

Therefore, we have used MIF to construct a cyber analytics pipelinedue to its ease of swapping in and out components, message types, and network protocols. Further,

[4] http://www.snort.org/

MIF's support for the graphical creation, execution, and modification of pipelines greatly eases the burden of building robust and reusable pipeline applications.This makes it easy to sendanalytic outputs to multiple complementary visualization tools that are needed in order to detect complex, emerging, and novel network attacks [8].

Our MIF-based cyber analysis pipeline is able to consume, transform, filter and analyze network flow data as it is produced in real time. The pipeline generates data for three different visualization tools for real time analysis.

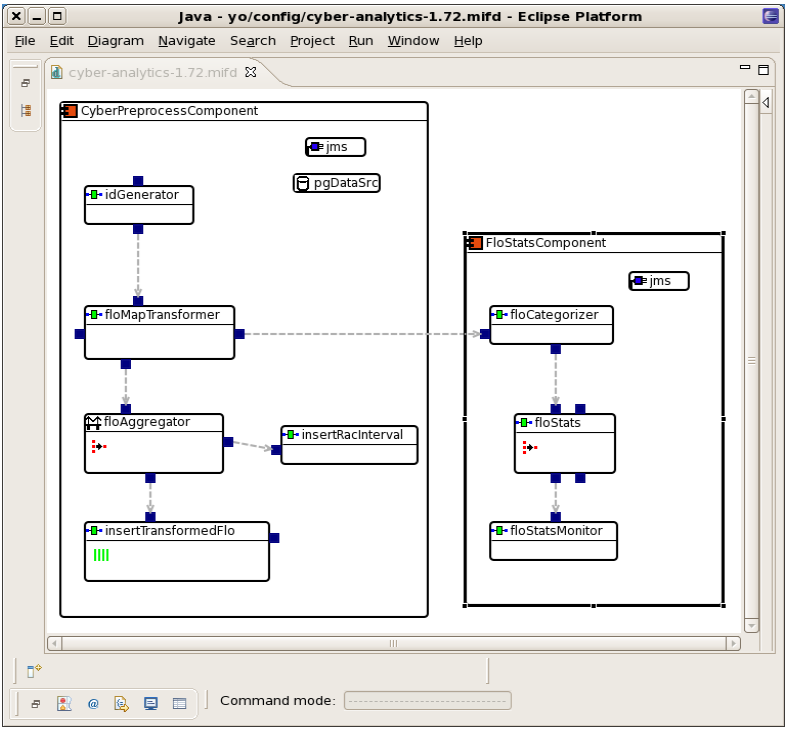


Fig. 6. Cyber pipeline design in MIF Component Builder

The pipeline, depicted in the MIF Component Builder in Figure6, comprises two high level components, namely *CyberPreprocessComponent* and *FloStatsComponent*. An external data ingest program (not shown) pushes data from network sensors over a JMS topic endpoint to the pipeline. *CyberPreprocessComponent* performs transformation, aggregation and loading of raw and aggregated data into the database, and outputs the data over another JMS topic for visualization displays to consume. The data is also sent to the *FloStatsComponent*, which categorizes the network data and attempts to fit the traffic patterns to one or more statistical distributions. These summary statistics are also sent over JMS for listening visualization tools to consume. If any incoming data is found to be anomalous, it is sent over a dedicated *event* topic, which triggers an alarm state in the visualization displays.

Performance tests were performed with MIF, the ActiveMQ JMS server, and a Postgres database. The MIF hostwas a dual quad core 2.80 GHz Intel Xeon® processor with 16 GB of memory.In testing with replayed network sensor inputs, the pipeline easily kept up with the average actual arrival rate of 83 flow msg/s and the peak of 145 msg/sWe then stress-tested the pipeline by progressively increasing the traffic replay rate well past real values. For the best run, the system reported a maximum average throughput of 2,781 msg/s, or over 240 million messages per day on a single node. The peak throughput 30-second interval was 3,350 msg/s. During this time, neither of the servers reached above 50% CPU utilization, leading us to conclude that communication between MIF components and the database wasthe bottleneck, and not the MIF pipeline execution.

6 Further Work and Conclusions

We are using MIF on several large projects in PNNL focusing on large-scale scientific data management and processing data from data streams. The technology is proving robust, fast and scalable. In addition, we are investigating using MIF as the core pipelining infrastructure for a petascale, federated collection of biology data. In this context, our work is focused on introducing adaptivity into MIF, so that based on policies or historical performance data, MIF pipelines can dynamically reconfigure, moving processing components to remote sites where the required data is located. Such a capability would enable us to more efficiently implement pipelines which process very large-scale data sets.

The complete MIF implementation is open source and freely available from http://medici.pnl.gov

References

1. Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M.: Pattern-Oriented Software Architecture: A System of Patterns, vol. 1. Wiley, Chichester (2009)
2. Yang, X., Bruin, R.P., Dove, M.T.: Developing an End-to-End Scientific Workflow. Computing in Science and Engineering, 52–61 (May/June 2010)
3. Shah, A.R., Singhal, M., Gibson, T.D., Sivaramakrishnan, C., Waters, K.M., Gorton, I.: An extensible, scalable architecture for managing bioinformatics data and analysis. In: IEEE 4th International Conference on e-Science, Indianapolis, Indiana, December 7-12, pp. 190–197. IEEE Computer Society, Los Alamitos (2008)
4. Gorton, I., Huang, Z., Chen, Y., Kalahar, B., Jin, S., Chavarria-Miranda, D., Baxter, D., Feo, J.: A High-Performance Hybrid Computing Approach to Massive Contingency Analysis in the Power Grid. In: Fifth IEEE International Conference on e-Science and Grid Computing, pp. 277–283. IEEE, Los Alamitos (2009)
5. Chase, J.M., Gorton, I., Sivaramakrishnan, C., Almquist, J.P., Wynne, A.S., Chin, G., Critchlow, T.J.: Kepler + MeDICi - Service-Oriented Scientific Workflow Applications. In: 2009 IEEE Congress on Services - Part I (Services-1 2009), pp. 275–282. IEEE, Los Alamitos (2009)

6. Barker, A., van Hemert, J.: Scientific Workflow: A Survey and Research Directions. In: Wyrzykowski, R., Dongarra, J., Karczewski, K., Wasniewski, J. (eds.) PPAM 2007. LNCS, vol. 4967, pp. 746–753. Springer, Heidelberg (2008)
7. Kiebel, G.R., Auberry, K.J., Jaitly, N., Clark, D., Monroe, M.E., Peterson, E.S., Tolic, N., Anderson, G.A., Smith, R.D.: PRISM: A Data Management System for High-Throughput Proteomics. Proteomics 6(6), 1783–1790 (2006)
8. Best, D.M., Bohn, S., Love, D., Wynne, A., Pike, W.A.: Real-time visualization of network behaviors for situational awareness. In: Proceedings of the Seventh international Symposium on Visualization For Cyber Security, VizSec 2010, Ottawa, Ontario, Canada, September 14, pp. 79–90. ACM, New York (2010)
9. Wilde, M., Foster, I., Iskra, K., Beckman, P., Zhang, Z., Espinosa, A., Hategan, M., Clifford, B., Raicu, I.: Parallel Scripting for Applications at the Petascale and Beyond. Computer 42(11) (2009)

Facilitating Enterprise Service Discovery for Non-technical Business Users

Marcus Roy, Basem Suleiman, and Ingo Weber

SAP Research, Sydney NSW 2060, Australia
School of Computer Science and Engineering, UNSW, Sydney NSW 2052, Australia
{m.roy,basem.suleiman}@sap.com, ingo.weber@cse.unsw.edu.au

Abstract. Enterprise Services (ES) are Web services with which enterprise applications expose a subset of their functionality. Due to the often high number of different ES, as well as the complex nature of their names, it is difficult for non-technical business users to discover services in ES repositories. However, most of this complexity stems from a SOA governance-driven service design process that is essential to the development of harmonized and long-lasting ES. Based on the example of SAP's ES, we describe a representational model that consolidates existing models and patterns used during the service design process. We created an iterative search approach that uses this consolidated metadata. The evaluation of the approach with real business users, based on a prototypical implementation, demonstrates that our iterative search is more efficient and effective than the currently offered search.

1 Introduction

Business process automation enables a cost-effective, agile and rapid composition and execution of new applications to address on-demand and changing organizational requirements. It leverages a business user's expertise to perform the modeling of business processes by specifying relevant activities that in turn are implemented by experienced developers reusing existing Enterprise Services (ES) in a Service-Oriented Architecture (SOA). In IT organizations, ES represent non-public Web Services intended for internal developers, partners and customers to reuse enterprise-specific data and functionality from existing legacy applications stored in company-internal repositories, e.g. SAP Enterprise Service Repository and Registry (ESR) or IBM WebSphere Service Registry and Repository (WSRR). They are idiosyncratically designed to represent specific parts of integrated and enterprise-internal business processes and legacy applications leading to long, technical and less comprehensive signatures (in this work we refer to service interface and operation names only) as shown in Examples (1.1- 1.3) of an "Sales Order" related SAP ES. These examples also underline the difficulty to discover ES as described by Beaton et al. [3,2]. Generally, when it comes to the naming of ES, it is a challenge to give unique and easy-to-understand names to a large number of ES, e.g. SAP Business Suite offers more than 4000 ES (08/2010). Most of its complexity has been introduced deliberately during its development life-cycle to ensure long-lasting, unique and easy-to-manage ES from a software

E.M. Maximilien et al. (Eds): ICSOC 2010 Workshops, LNCS 6568, pp. 100–110, 2011.

engineering perspective. For instance, the ES shown in Example (1.1) is quite detailed and less prone to duplication. It describes a read operation of a Sales Order, stored in an ERP system, which is identified by an ID expected as part of the request. Most of these intrinsic characteristics are inferred from a standardized service design process applied prior to the service implementation as part of a SOA governance process that guides and governs the development of ES. It is indispensable for large enterprises to follow such a globally agreed on SOA governance process.

`SalesOrderERPByIDQueryResponse_In` (1.1)

`SalesOrderCreateRequestConfirmation_In` (1.2)

`SalesOrderBasicDataByBuyerAndBasicDataQueryResponse_In` (1.3)

In the following, we used the example of an SAP service design process to illustrate the use of agreed-on methodologies and proprietary models that can be adopted to what we call a *service* and *data model*. Principally, such a design process is not limited to SAP, but can also be found with others companies (not the focus here). We then introduced a representational model that integrates both service and data models as first-class components including their relationship. Based on this model, we created a metadata repository based on a list of ES and their respective metadata that has been extracted from multiple sources, e.g. corporate web pages and semi-structured documents. This metadata hereby refers to entities in the data and service model respectively. Note that the focus of this paper is not on yet another service description technology. We are not competing with existing technologies, e.g. RDF or OWL-S, instead we could utilize them to represent such metadata as annotations. We then implemented the conceptual solution of our iterative search in form of a prototype. We performed a first evaluation of this prototype with real business users, which demonstrates the feasibility of our approach to enhance the current search for ES.

In the remainder, we explain the service design process in Section 2 and introduce our representational model in Section 3. Section 4 details on our iterative search and prototype. In Section 5, we evaluate our search and prototype, refer to related work in Section 6 and conclude the paper in Section 7.

2 Service Design Process

In this section, we describe the application of a design process for the service development. Based on the example of SAP's ES, we detail on the existing data and service model used to develop ES as shown in Examples 1.1-1.3.

In the introduction, we generally referred to a SOA governance-driven service design process [6,1] that is deployed to guide and ensure the development of harmonized and business-aligned ES. It is also used to create a mutual understanding about the definition and meaning of ES among stakeholders and partners. Such a design process is required and executed recurrently prior to any definition and implementation phase. Using the example of an SAP service

design process, developers are guided in their tasks to first design ES by specifying high-level and business-related concepts that abstractly describe the purpose and scope of the prospective ES.

Most of these design efforts are supported by a modeling tool to visually compose an abstract service definition as shown in Figure 1. Such an application enables developers to arrange and order abstract shapes representing existing key business entities (red shape) as well as the preferred but predefined way of accessing them (blue shape). Eventually, this visual representation is transformed into an abstract service definition, i.e. coarse WSDL template, detailing only on service interfaces and operations.

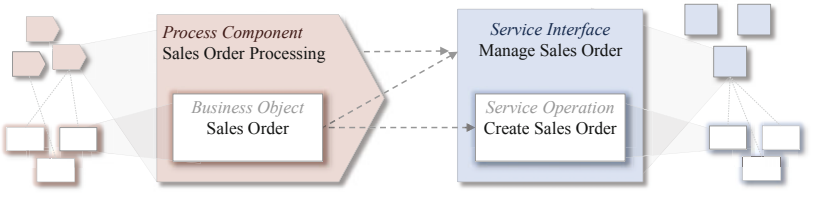


Fig. 1. Visual model of an abstract service definition during service design

Although a modeling tool helps to visually generate an abstract ES definition based on entities from the data and service model, this information is typically not available to a search environment (though it's hidden in documentation). Therefore, Figure 2 illustrates earlier mentioned models, i.e. data and service model, and their relationship, which embraces key entities of information that can me made available to a search. For simplicity's sake, we only focused on showing a relevant subset of model entities that are considered significant for the definition of service interface and operation names.

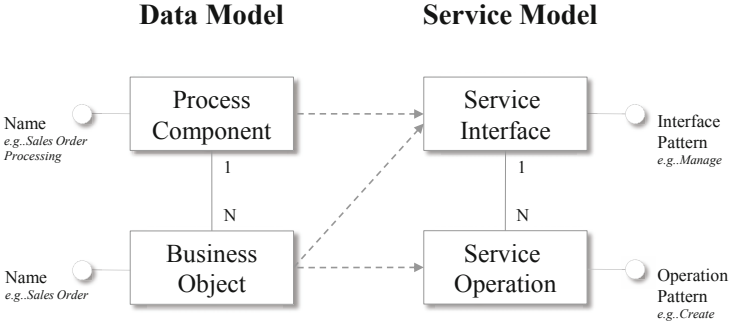


Fig. 2. Subset of the data and service model shown as ER diagram

In Figure 2, a Process Component "Sales Order Processing" represents a technical concept similar to a department inside a company that groups all Sales Order related activities (e.g. Create, Approve, Notify etc.). Within such a Process Component, a Sales Order Business Object denotes a central business entity

Table 1. Construction of the service interface name

Service Interface	
Process Component	`Sales Order Processing`
Business Object	`Sales Order`
Interface Pattern	`Manage`
Service Interface	`SalesOrderProcessingManageSalesOrder`

Table 2. Construction of the service operation name

Service Operation	
Business Object	`Sales Order`
Interface Pattern	`Create`
Service Operation	`SalesOrderCreateRequestConfirmation_In`

that stores and maintains all relevant information about a real Sales Order (e.g. Buyer, Seller etc.). Figure 2 also shows service model entities, i.e. Service Interface and Operation, that are associated to related data model entities, i.e. Process Component and/or Business Object. Based on both model entities and their association, potential signatures for service interface (Table 1) and operation (Table 2) are defined. In Table 1, the service interface implicitly describes a grouping of related service operations that manage a Sales Order of a Sales Order Processing activity.

The use of the keyword "manage" in the service interface signature is based on an interface pattern (as shown in Fig. 2) that determines selective service operations. Hence, service operations listed under a "manage" interface have to be either a "create", "read", "update", "change", "check" or "cancel" operation (called *operation pattern*). In Table 2, a service operation is defined based on a "Sales Order" Business Object and the "create" operation pattern, which is implied by the "manage" interface pattern of Table 1.

3 Leveraging Service Design Principles

The example of the previous section illustrated the design of an abstract definition for service interface and operation names based on the service and data model model. In this section, we describe how to utilize the knowledge of this service design process. Firstly, we designed a representational model (Section 3.1) by consolidating the data and service model. Second, we created a metadata repository defined by our representational model and populated it with information extracted from multiple sources (Section 3.2). Both representational model and metadata repository represent the basis of the iterative search as explained in Section 4.

3.1 A Representational Model

All information stored in our metadata repository is defined according to a representational model consisting of the data model extended with entities of the

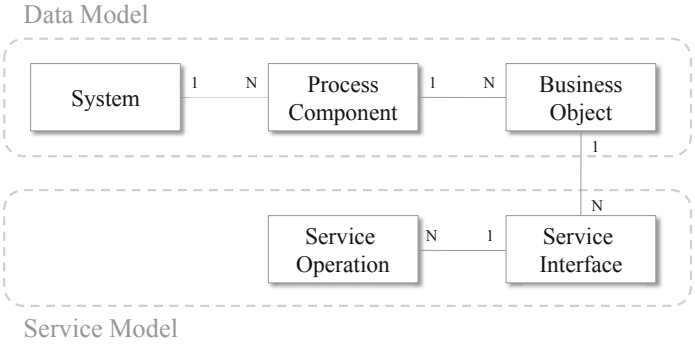


Fig. 3. Enterprise Service Representation Model

service model. Originally, both models are used in rather different context meaning that the knowledge about the relationship between both models is not given by default, and thus, not obvious to anyone trying to discover ES. We therefore consolidated both models by integrating them and describe their relationship as the association of Business Object to Service Interface (Fig. 3). Having both models and their relationship defined as first-class components in our representational model has the advantage of naturally exploiting services based on their underlying data model and vice versa. This means we can describe a service based on its concrete relation to specific data model entities or in turn find any services that are related to a particular data model entity.

3.2 Extraction of Service Design Entities

In order to create the basis for our ES Search approach, which we present in the next section, we created a metadata repository that represents a list of all available SAP ES that can be found on SAP's Enterprise Service Workplace[1] (ESW). The ESW is a central place used by developers and consultants to search for and view detailed information about ES. It includes documentation, examples, and technical descriptions, e.g. WSDL.

Starting from this list of ES, we enriched the entries with additional metadata according to our representation model, by extracting information from multiple sources as follows. Firstly, we extracted data model-related information from a set of reports (e.g. Excel spreadsheets) that were exported from an operational content management system. Secondly, we extracted service model data directly from the ESW: we used simple page scraping techniques to extract data from the web site of each Enterprise Service.

We then used basic inference techniques and regular expressions on the extracted data, so as to detect model entities in the service interface and operation signature. Subsequently we mapped the respective service and service model entities to the data model based on relationship as defined in our representation

[1] `http://www.sdn.sap.com/irj/bpx/esworkplace`

model and created the association in our metadata repository. We also recognized interface and operation patterns and annotated the corresponding service model entities.

4 Enterprise Service Search

In this section, we describe the concept of the iterative search and its prototypical implementation. The search uses the repository from the previous section.

4.1 Iterative Search Approach

In order to significantly improve the search, we first analyzed how users search and what they enter to find a particular ES. This analysis has been conducted on anonymized log files of search queries entered by users over a period of six months. In detail, the examined log files contained search strings that have been categorized according to entities in our model, if applicable. Finally, we accumulated recurrent categories to rank their frequency of occurrence (not in scope of this work). One key finding was that users rarely enter more than three keywords. Secondly, they frequently entered simple keywords representing central business entities that formed the basis of the desired ES, e.g. "Sales Order" for a "Create Sales Order" ES etc. Based on these observations, we created an iterative search that allows users to choose from a set of predefined search options rather than allowing them to freely enter search text. Each search option conceptually relates to entities in our model whereas each selection of a search option identifies a node in the metadata repository. Starting from this node, a list of respective ES can be inferred by following up on any consecutive parent-child-relationship down to the Service Operation level as illustrated in Figure 4.

Using such a search-by-selection technique better leverages the underlying data model from a twofold perspective. Firstly, potential search options (as illustrated in Figure 4) can be directly connected to related entities in our representation model without the need to understand and map a potentially volatile human search text. Secondly, we can use our metadata repository to populate search options from which the user can choose. With each selection, the content of any consecutive search option is determined while the list of potential ES candidates gets iteratively filtered. In the end, the more information is provided by the user, the more effective is the filtering and the smaller is the result set of potential ES (enabling a subsequent browsing as a final search step). Our evaluation in Section 5 demonstrates the effectiveness and efficiency if this approach.

4.2 Search Prototype and Example

We have implemented the iterative search as a Web application (cf. Figure 4) that uses our representational model and metadata repository. In the top half of the application (Part A), the user can choose from previously mentioned search options. For each selection, a list of potential ES will be updated and displayed in the bottom part of the application (Part B). Users can skip search options if they are unsure about their significance or if the search is already successful.

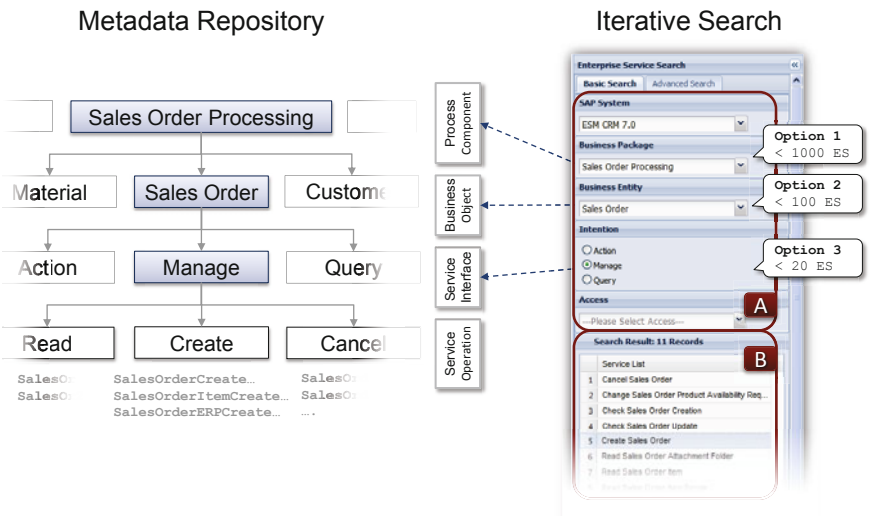


Fig. 4. Correlation of our representational model to iterative search prototype

As an example, say a business user wants to find an Enterprise Service that "creates a Sales Order". First and without any selection, a complete list of service operations is shown (> 4000 ES). In a next step, the user selects "Sales Order Processing" from a list of available Process Components (Option 1/Figure 4), which decrements the result by an order of magnitude as only ES are displayed that belong to that particular department. In a second step, the user selects a "Sales Order" from a list of Business Objects determined by the previously chosen Process Component (Option 2/Figure 4), which reduces the result set by another order of magnitude. By that time, users are already able to browse the significant smaller list of ES to find possible matches. In the case of uncertainty, the user can continue to choose from the remaining set of options by detailing on the intention what to do with the Sales Order. The intention to create a Sales Order implies a "manage" pattern (Option 3/Figure 4), which reduces the search result to 11 ES as partly shown in Part B of Figure 4. This search result correctly contains the desired `SalesOrderCreateRequestConfirmation_In` ES, which has been given the synonym "Create Sales Order" for reasons of readability.

5 Evaluation

We evaluated our discovery approach by exposing the above-described iterative search prototype to a group of test users, as discussed in this section. Since the time for searching and displaying relevant ES only takes 1-2 seconds, performance seems to be no big concern in our approach and has not been evaluated further.

5.1 Experimental Setup

Five business users from SAP business consulting and support departments have been briefly introduced to the ESW search and our iterative search prototype.

The participants had never used either one of them. They also have been given a four-step "Create Sales Order" business process (see Fig. 5) with brief description of the process activities. They then have been asked to find the relevant ES for each activity of the given business process by using (1) our iterative search prototype and (2) the ESW – in that order, so the iterative search would not benefit from a possible learning curve. Each participant was given 30 minutes to complete the task.

Fig. 5. "Create Sales Order" Business Process used in the experiment

5.2 Data Collection and Analysis

The participants notified us whenever they thought they found a service for one of the activities. During each experiment (1) and (2) for each of the five participants, we recorded the time spent to find each ES. We also observed the participants' behavior in terms of how easy or hard it was for them to use the respective application to find the relevant services. After each experiment, we asked the participants about their experience with each of the investigated tools. We recorded all their feedback for future improvements. The summary of the results collected from our experiments are grouped into two main categories: First, the search time represents the overall time (in seconds) spent by each participant to find all relevant ES and second, the total number of correct ES found by each participant (out of 4 services).

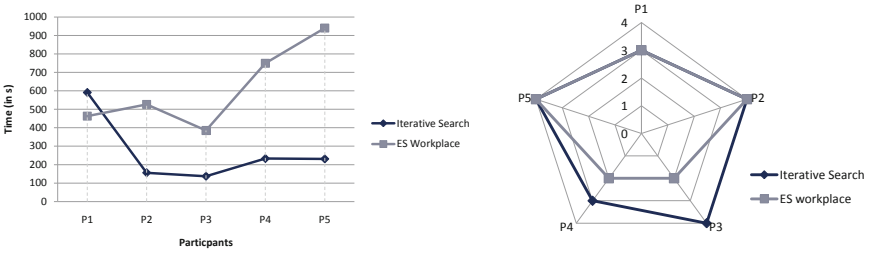


Fig. 6. Evaluation results, per participant, using the ESW vs. our iterative search. **Left:** search time; **Right:** number of correctly discovered services.

The result, as depicted in Fig. 6 (left) , shows that the search time spent by almost all participants, except for participant P1, using our iterative search prototype is noticeably less than the time required by the same participants using the ESW. For instance, participants P4 and P5 spent only about a third of the ESW search time using our iterative search to find the four ES. With respect to participant P1, we noticed some degree of uncertainty when using our prototype, compared to the other participants. We also noticed that he

found some ES on the ESW by coincidence. In summary, the average search time spent by all participants using our iterative search is more than half of the time spent by the same participants using the ESW. Although the number of asked participants is not significant large, the results still clearly indicate an improvement of our iterative search approach over the existing ESW in terms of search time.

The second evaluation criterion is the number of correct ES found by participants using both search applications. As shown in Fig. 6 (right), 60% of participants were able to find all four correct ES by using our iterative search prototype compared to 40% of participants using the ESW search. The minimum number of ES found is two on ESW vs. three with our iterative search. Although 40% of the users have not found *all* four correct services (sometimes as simple as a "Create Sales Order"), they at least found three out of four ES and therefore did not completely fail. On the basis of these initial results, we can say that our approach has the potential to iteratively improve the search, with room for further improvements. Although the number of five users can be considered marginal to comprehensively validate our search, we believe it yet has shown the potential to improve the discovery of Enterprise Services over the existing search.

6 Related Work

In order to position our work, we classified current service retrieval techniques into linguistic, semantic and hybrid approaches.

Linguistic-based approaches are mainly based on textual analysis of service description such as a WSDL using clustering such as [5,8,17] techniques. For instance, the Woogle [5] search engine uses a similarity search for operations by grouping operation parameters into meaningful concepts.

Semantic approaches are based on extending service descriptions with formal models to capture their underlying meaning. The resulting so-called Semantic Web Services (SWS) [11] relate service properties to concepts belonging to ontologies – roughly speaking, formal conceptual models of the terms in a given domain, including relationships and constraints between terms. The discovery approach is usually to match concepts of queries against SWS descriptions, using logic-based inference. In [4], the service discovery algorithm matches service request against the services ontology using description logic. Similarly, [10,12] present service matchmaking approaches to enable discovery of ontology-based service descriptions based on DAML-S. The matching technique in [9] relates service inputs and outputs using OWL ontologies to capture the meaning of terms in service descriptions. Also, [14] describes the formalization of a client query as a goal that is matched to the goal of a service. However, semantic discovery requires (i) an agreed-upon common ontology, (ii) complete semantic service descriptions, which can require considerable manual effort during service development, and (iii) semantic models for queries.

Hybrid approaches, e.g. [13,15,18,7,16] use the ontology and semantic techniques to enhance linguistic service discovery. For example, the Seekda [13] public

search engine relies on crawling and data mining techniques of Web API repositories such as ProgrammableWeb (http://www.programmableweb.com/) in order to automate the annotation of APIs with metadata that can be used for a semantic search. The Opossum [16] search engine also crawls the Web for WSDL descriptions and transforms them into a generic ontological-based model. The service properties are then automatically enriched with concepts from existing ontologies of different domains (e.g. finance, e-commerce). Search is done by matching concepts of the query against the ontological-based model of the service. This is done by considering the linguistic and structural properties of ontology.

Our approach is different from the above since we base it on concepts that stem from an organizational SOA governance-driven service design process which guides the development of the ES. In a way, the collection of terms that are instances of the data and service model elements can be seen as a specialized ontology. The corresponding vocabulary contains rather business terms than technical concepts. As such, discovery in this context cannot be put on a par with discovery of arbitrary Web services on the Web. Instead, ES are very much predictable as they comply to internal design principles, which are created once and applied to all existing and future ES.

7 Conclusion and Future Work

In this paper, we investigated how SOA governance artifacts can be used during service discovery. We first explained why governance is necessary when large amounts of services are developed, and showcased the artifacts that can result from governance at the example of the more than 4000 Enterprise Services from SAP. At the core of the information of interest in this context are the *service* and *data models*, which are used to abstractly define ES interface and operation names. For our discovery approach, we created a representational model offering a consolidated view on both models. Based on this combined model and its associations, we created a metadata repository of all SAP ES, and enriched it with the metadata extracted from multiple (real-world) sources. We then developed an iterative search based on this metadata repository. An initial evaluation of our prototype with a group of test users demonstrated the feasibility and effectiveness of our approach.

In the future, we plan to more generally investigate the complex nature of governance-driven service design. We believe this will lead to richer ES descriptions, and ultimately a more sophisticated and effective search technique. Finally, we will continue evaluate our techniques with real users as the work progresses.

References

1. Artus, D.J.: SOA Realization: Service Design Principles (February 2006), http://www.ibm.com/developerworks/webservices/library/ws-soa-design/
2. Beaton, J., Jeong, S.Y., Xie, Y., Stylos, J., Myers, B.A.: Usability Challenges for Enterprise Service-oriented Architecture apis. In: VLHCC, IEEE, Los Alamitos (2008)

3. Beaton, J.K., Myers, B.A., Stylos, J., Jeong, S.Y.S., Xie, Y.C.: Usability evaluation for enterprise SOA APIs. In: SDSOA 2008, ACM, New York (2008)
4. Benatallah, B., Hacid, M., Leger, A., Rey, C., Toumani, F.: On Automating Web Services Discovery. VLDB Journal 14(1), 84–96 (2005)
5. Dong, X., Halevy, A., Madhavan, J., Nemes, E., Zhang, J.: Similarity Search for Web Services. In: VLDB 2004: Proceedings of the Thirteenth International Conference on Very large Data Bases, pp. 372–383. VLDB Endowment (2004)
6. Erl, T.: Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall Professional Technical Reference (2005)
7. Fenza, G., Loia, V., Senatore, S.: A Hybrid Approach to Semantic Web Services matchmaking. Int. J. Approx. Reasoning 48(3), 808–828 (2008)
8. Funk, A., Bontcheva, K.: Ontology-based Categorization of Web Services with Machine Learning. In: LREC 2010, Valletta, Malta, ELRA (May 2010)
9. Hull, D., Zolin, E., Bovykin, A., Horrocks, I., Sattler, U., Stevens, R.: Deciding Semantic Matching of Stateless Services. In: AAAI (2006)
10. Li, L., Horrocks, I.: A Software Framework for Matchmaking based on Semantic Web Technology. In: WWW 2003 (2003)
11. McIlraith, S.A., Son, T.C., Zeng, H.: Semantic Web Services. IEEE Intelligent Systems 16(2), 46–53 (2001)
12. Paolucci, M., Kawamura, T., Payne, T., Sycara, K.: Semantic Matching of Web Service Capabilities. In: Horrocks, I., Hendler, J. (eds.) ISWC 2002. LNCS, vol. 2342, p. 333. Springer, Heidelberg (2002)
13. Steinmetz, N., Lausen, H., Brunner, M.: Web service search on large scale. In: Baresi, L., Chi, C.-H., Suzuki, J. (eds.) ICSOC-ServiceWave 2009. LNCS, vol. 5900, pp. 437–444. Springer, Heidelberg (2009)
14. Stollberg, M., Keller, U., Lausen, H., Heymans, S.: Two-Phase Web Service Discovery based on Rich Functional Descriptions. In: Franconi, E., Kifer, M., May, W. (eds.) ESWC 2007. LNCS, vol. 4519, pp. 99–113. Springer, Heidelberg (2007)
15. Toch, E., Gal, A., Reinhartz-Berger, I., Dori, D.: A Semantic Approach to Approximate Service Retrieval. ACM Trans. Inter. Tech. 8(1), 2 (2007)
16. Toch, E., Reinhartz-Berger, I., Gal, A., Dori, D.: $\mathcal{OPOSSUM}$: Bridging the Gap Between Web Services and the Semantic Web. In: Etzion, O., Kuflik, T., Motro, A. (eds.) NGITS 2006. LNCS, vol. 4032, pp. 357–358. Springer, Heidelberg (2006)
17. Wu, J., Wu, Z.: Similarity-based Web Service Matchmaking. In: SCC 2005, Washington, DC, USA, pp. 287–294. IEEE Computer Society, Los Alamitos (2005)
18. Zhang, Y., Liu, B.-Y., Wang, H.: A Method of Web Service Discovery Based on Semantic Message Bipartite Matching for Remote Medical System. J. Theor. Appl. Electron. Commer. Res. 4(2), 79–87 (2009)

Hypermedia-Driven RESTful Service Composition

Rosa Alarcon[1], Erik Wilde[2], and Jesus Bellido[1]

[1] Computer Science Department
Pontificia Universidad Catolica de Chile
ralarcon@ing.puc.cl, jbellido@uc.cl
[2] School of Information
UC Berkeley
dret@berkeley.edu

Abstract. *Representational State Transfer* (REST) services are gaining momentum as a lightweight approach for the provision of services on the Web. Unlike WSDL-based services, in REST the set of operations is reduced, standardized, with well known semantics, and changes the resource's state. Few attempts have been proposed to support composition models for REST, they are mainly operation-centric and fail to acknowledge the hypermedia nature of REST, that is, clients must inspect the served resource state and choose the link to follow from there. We explore RESTful service composition as it is driven by the hypermedia net that is dynamically created while a client interacts with a server resulting in a light-weight approach. We based our proposal on a hypermedia-centric REST service description, the *Resource Linking Language (ReLL)* and Petri Nets as a mechanism for describing the machine-client navigation.

1 Introduction

SOA services provide an endpoint that exposes a set of *operations* on entities that are out of the reach of clients. Operations are described in a standard WSDL document; semantics are not explicit and are usually specified in additional documents so that client designers understand the scope, effects, pre-conditions and assumptions made by service designers and program the clients accordingly. Clients interact with servers following the description (they are *tightly coupled*), if it changes clients fail, clients cannot be notified about changes and failure semantics and its recovery are ad-hoc. Client-server interaction state is kept by the server (*stateful*), which negatively impacts service scalability and increases the complexity of coarse grained operations.

The REST architectural style has been characterized as a restricted subset of SOA [1]. Unlike WSDL operations, in REST the central elements are the *resources*, which are abstract entities identified by URIs that can be manipulated through a *uniform interface*, that is, a reduced set of standard operations whose semantics are well known in advance and are defined by standard transport protocols such as HTTP. Resource's *state* is transfered to/from the clients as a

E.M. Maximilien et al. (Eds): ICSOC 2010 Workshops, LNCS 6568, pp. 111–120, 2011.

consequence of executing the standard operations. The state is portrayed to the clients by means of *representations*, which are documents serialized according to specific media types (e.g. XML), and contain hyperlinks to related resources, and *controls* that allow clients to perform operations and change resources' state (e.g. `<link=URI rel="service.POST">`, `<form ...>`, etc.). There is no guarantee that the operations, the resources or even the network remain available or unchanged, however, there is a uniform failure interface (i.e. standard protocol error codes) with well known semantics that allow clients to recover accordingly.

A REST service is not an endpoint but a web of interconnected resources, with an underlying hypermedia model that determines not only the relationships among resources but also the possible net of resource state transitions. REST clients discover and decide which links/controls to follow/execute at runtime. This constraint is known as HATEOAS (Hypermedia As The Engine Of Application State). Hence, it is possible to provide a single or a small set of URIs as *entry-points* to the whole service web or a subset of it. REST services consider humans as its principal consumer and they are expected to drive resource discovery and state transition by understanding the representation content. The lack of a machine-readable description forces REST service providers to describe their APIs in natural language which makes difficult to properly design machine-clients and recently is being studied as an infrastructure layer for supporting service composition and business processes. Recent proposals, however, heavily rely on the *operation*-based model neglecting the hypermedia characteristics of REST. In this paper, we explore the impact of the hypermedia (HATEOAS) property for supporting machine-clients that implement RESTful service composition. Unlike current approaches, we based our strategy on a service description called ReLL [2], that is also based on the hypermedia property. The approach allow us to implement a machine client that is able to perform dynamic discovery of REST resources.

This paper is organized as follows, Section 2 present related work in both REST composition and REST description; Section 3 briefly presents ReLL, the Resource Linking Language for REST service description; Section 4 introduces an example to illustrate a composition model and language based on Petri Nets; and Section 5 presents conclusions and future work.

2 Related Work

In [3] composition requirements specific to REST services are identified, such as, *dynamic late binding*, that is the resource's URI to be consumed is known only in run-time; the composition technique must support the REST *uniform interface*; *dynamic typing*, methods may require parameters with types known only on run-time; *content type-negotiation*, the representations' media type for the component services as well as the composed service can be negotiated; and clients should be able to *inspect the state* of the composition.

JOpera [3] satisfies such requirements, and it is one of the most mature platforms for supporting REST services composition. JOpera provides a visual language for defining a *control flow* and a *data flow transfer* graphs, as well as

an execution engine for the resulting workflow. Nodes in the control flow graph represent *tasks* that are dynamically bound to *adapters* such as local UNIX programs, services invocation, etc., and "glue" adapters that perform local computations (e.g. XPath queries, XSLT transformations, etc.). Tasks and adapters have input and output parameters, for instance, HTTP adapter parameters are: `Method`, `URI`, `Body` and headers (`headin`). Adapter invocation order is regulated by *control tasks* that define conditional synchronization points, conditional loops, and forks. The data flow graph makes explicit the data flow between tasks and the resulting composition is written as a BPEL extension for REST [4].

Similarly, Bite [5] proposes a BPEL-inspired workflow composition language describing both control and data flow. Bite partially supports an HTTP-based uniform interface, dynamic typing, state inspection (both `GET` and `POST`). Regarding dynamic late binding, Bite can mint URIs for resources created, but can not inspect representation content and selectively retrieve the URIs served by the service. In both JOpera and Bite, the composition workflow is seen as a unique composed resource. In [6], it is possible to inspect the state of the workflow instance (i.e. the composed resource or *case*) and the tasks that compose it. State transition is modeled as links following a *URI template* that can change dynamically. Tasks progression is driven by humans and no automatic support is provided to retrieve and follow the links embedded in the representation.

Decker [7], presents a formal model for REST process enactment based on Petri Nets, PNML (Petri Net Markup Language), and an execution engine. They partially support dynamic late binding by minting URIs for created resources but do not support the HATEOAS constraint, neither complex guard conditions such as authentication, nor content negotiation (only XML as media type).

On the other hand, it is argued that a service description introduces a contract between clients and servers and hence some degree of coupling. In practice, service developers provide informal documents for its REST APIs describing the resource types, the set of *entry points* (static URIs) and URI patterns for the resources, authentication mechanisms, protocols, operations and media types, and even representation content samples, so that machine-clients can be designed accordingly. Documents may end up outdated, inaccurate or unclear, forcing developers to engage in trial-error phases to accommodate such changes.

A few languages have been proposed to create RESTful services description. For instance, the Web Application Description Language (WADL) [8] describes RESTful services as *resources* identified by URI patterns, media types and the schemas of the expected *request* and *response*. Representations support parameters that can contain links to another resources. WADL does not support link discovery or link generation for new resources, the resulting model is operation-centric and introduces additional complexity with unclear benefits for both human and machine-clients. Other proposals such as WRDL (Web Resource Description Language), and WDL (Web Description Language) [4], introduce less complex descriptions to model resources, but they focus also on the operations allowed for each particular resource and their input and output parameters and do not support the HATEOAS property.

3 Resource Linking Language (ReLL)

In [2] we introduced ReLL, the *Resource Linking Language*, which describes REST services with emphasis on the hypermedia characteristics of the model. Figure 1 presents the metamodel that comprises the main constraints of REST and is the basis for ReLL. A ReLL XML Schema has been also produced [2] and is used to write ReLL XML descriptions. A snippet, written in ReLL, describing a `requestToken` resource is shown in Figure 2. A REST `service` exposes a set of one or more `resources` each with a unique identifier (`xml:id`), names and descriptions (human-readable labels) and optionally constraints for the expected resources, such as a `uri` pattern for the expected (`match`) resource URI (so that a machine client can identify a URI change). A resource may have multiple `representations`, which are the serialization of the resource in some syntax or media `type`. Representations can define schemas for validation of input data.

Each representation can contain any number of `links` that can be retrieved through `selectors` written in a language (`selector type`) that suits the representation media type. For instance, XPath (*XML Path Language*) expressions for XML-based representations since they allow structured selections within XML document trees. Selectors have a `name` and refer also to a `location` in the representation (e.g. the content or the metadata such as HTTP headers). Links relate resource's representations with other *resource type* (instead of a resource URI) as indicated by the `target`, in order to avoid coupling with the resources' naming scheme. A link can also specify a *protocol* and it is possible to mint or `generate a URI` by executing an expression (e.g. a concatenation written in XPath). ReLL allows also to annotate resources and links with `types`, so that application domain semantics can be explicitly declared without requiring changes in the existent resources.

A machine-client can use the description to automatically and selectively retrieve the underlying web of resources. For instance, in [9], RESTler, a Web crawler uses ReLL descriptions to retrieve resources from a Web site, and REST APIs (Twitter, and Google Calendar). Since domain semantics are explicitly supported, it is possible to transform the retrieved resources to its *semantic*

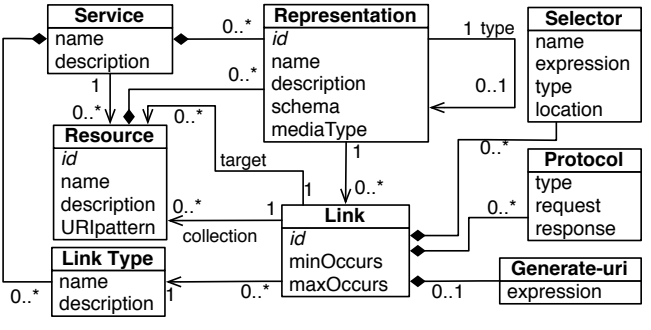


Fig. 1. ReLL Metamodel

```
<resource xml:id="requestToken">
    <uri match="https://api.linkedin.com/uas/oauth/requestToken" type="regex"/>
    <representation xml:id="requestToken-text" type="iana:text/plain">
        <name>oauth_token request parameters</name>
        <link xml:id="authorizeRq" type="request" target="authorize" minOccurs="0" maxOccurs="1">
          <selector name="oauthUrl" select="oauth_request_auth_url=[a-zA-Z0-9\-\%\.]+" type="regex"/>
          <selector name="oauthToken" select="oauth_token=[a-zA-Z0-9\-]+" type="regex"/>
          <generate-uri xpath="concat($oauth_url,'?',$oauth_token)"/>
          <protocol type="http">
              <request method="get"/>
              <response media="iana:text/plain"/>
          </protocol>
        </link>
    </representation>
</resource>
<resource xml:id="authorize">
     <uri match="https://api.linkedin.com/uas/oauth/authorize\?oauth_token=[a-zA-Z0-9\-]*" type="regex"/>
 </resource>
```

Fig. 2. LinkedIn ReLL snippet

counterpart through XSLT, and integrate the services at the semantic level. For instance, in [10] resources from four REST services (Flickr, Twitter, a Web site and a User mapping) are transformed to RDF, and integrated so that SPARQL queries can consider the resulting graph.

4 REST Service Composition

4.1 OAuth, LinkedIn and Facebook, a Composition Example

We illustrate these ideas by composing two REST services, LinkedIn and Facebook. A machine-client will retrieve the `SocialNetwork` (contacts and friends) from both sources and will provide an XML representation of the merged data. Entities (top rectangles in each thread) represent REST resources. Both services require user authentication based on the *OAuth* protocol, though with different versions, 1.0 for LinkedIn (Figure 3a) and 2.0 for Facebook (Figure 3b). This difference imply variations in the client-server conversation as well as the passed values. Client-server interaction occurs between the machine-client and the resources implementing the steps of the protocol (e.g. generate a *requestToken*, *authorize*, and generate an *accessToken*). Part of the interaction occurs out of band, as a separate conversation between the service provider and the user. Some parameters are sent to/from the server in the message body, others in the Headers, some messages must be signed, and some parameters are used as inputs for the next client-server interaction step.

The HATEOAS constraint is extensively used in Web content (i.e. it contains embed hyperlinks and controls) and is in great part responsible for the Web popularity since it allows users to dynamically discover material they are interesting in, but unfortunately it is not considered yet by REST API designers, that is, representations typically do not include hyperlinks but data fields that serve to generate or mint the URIs to follow. This practice introduces a strong coupling between clients and servers and forces machine-clients to mint their own URIs.

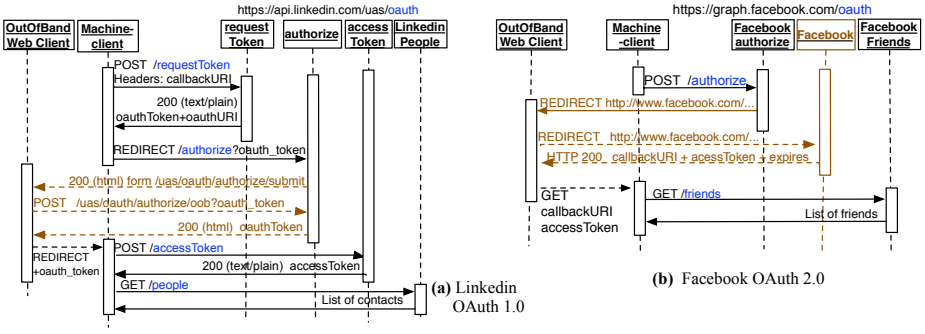


Fig. 3. OAuth sequence diagram for LinkedIn (a) and Facebook (b)

4.2 A Petri Net Model

We are interested in the development of machine-clients that enable service composition, so that B2B or mashups involving REST services could be easily developed. The HATEOAS constraint becomes fundamental provided that machine-clients can understand the semantics of the links and controls served in the representations. A ReLL document describes in a declarative way a partial (or whole) view of the web of resources, its domain semantics and the mechanisms required to navigate across such web, and hence, provides machine-clients access to a basic semantic model at a protocol and application level.

In [7], a formalization for *service nets* implementing RESTful processes is introduced. A service net is a colored Petri Net represented by a tuple $S = (\mathcal{P}, \mathcal{T}, \mathcal{F}, \mathcal{F}_{read}, \mathcal{T}_S, \mathcal{T}_R, init, g, uri)$, where $\mathcal{P}$ and $\mathcal{T}$ are disjoint sets of *places* and *transitions*. Places represent states that contain tokens with multiple attributes, and transitions represent activities that can be guarded; transitions are fired when all the tokens in the corresponding input place arrive. Places and transitions are connected through arcs. $\mathcal{F}$ represents the set of flows, and $\mathcal{F}_{read}$ the set of read arcs (GET). $\mathcal{T}_S, \mathcal{T}_R$ correspond to the disjoint sets of *send* and *receive* transitions (also called communication transitions), *init* is the initial marking, that is, the function that initially assigns multiple tokens (with values serialized as XML documents) to places, g is a function that assigns guard conditions to transitions and conditions are combinations of XML serialized input documents, and *uri* is a function that assigns URIs to tuples of communication transitions $(\mathcal{T}_S, \mathcal{T}_R)$ and combinations of XML serialized input documents, this feature allows the model to generate URIs. According to Decker, REST service composition is defined as the merge of *send* and *receive* transitions offered by senders and receivers that belong to separate service nets. One token is assigned to an input place and removed to an output place when a transition fires. In the case of read arcs, tokens are not removed from input places and there is no functional dependency between tokens.

Dynamic late binding in this model, is implemented by receive transitions $(\mathcal{T}_R)$ that take input tokens to generate new URIs called here dynamic ports. The rules for uploading information into tokens and for generating new URIs

are not presented, but this is not a trivial task since it may require parsing information, encrypting, digital signatures, store information into headers, etc.

This feature is fundamental in real life scenarios, as is shown by a popular language used for service composition in the industry, namely BPEL, which supports data transformation by means of XPath and XQuery expressions; or JOpera which uses a library of extensible adapters to satisfy the same requirement. The other aspect of *dynamic late binding*, that is, to inspect representation's content and determine from there the URI of a send transition (i.e. the HATEOAS constraint) is not discussed.

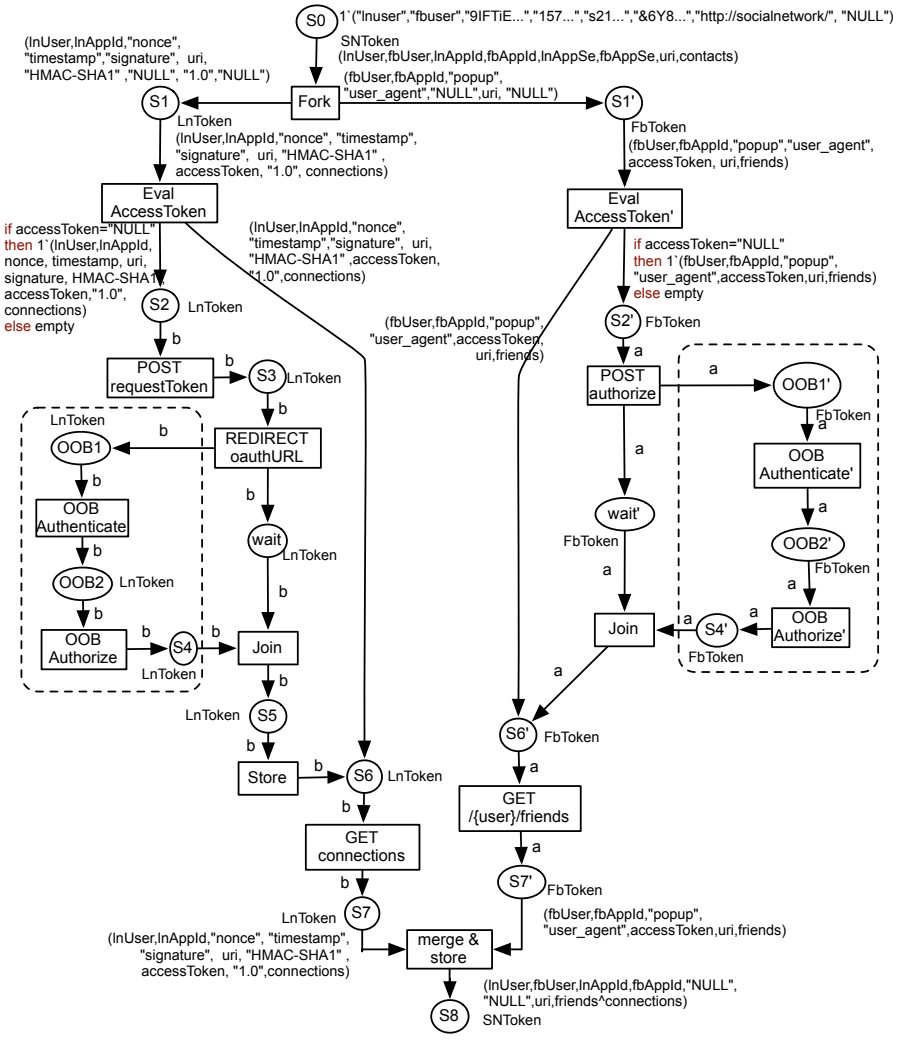


Fig. 4. A PetriNet for the composed resource: `socialNetwork`

The colored Petri Net shown in Figure 4 corresponds to the example of Section 4.1. It implements a `socialNetwork` resource that composes resources from LinkedIn and Facebook. The Petri Net was modeled and simulated using a CPN tool. Places (circles) represent states of the composed resource; tokens (set of attributes) are shown near places (e.g. LnToken); places receive a determined type of token and transitions provide the mapping between tokens with different attributes. Places and transitions within dotted rectangles and labeled as `OOBn` (Out Of Band) and refer to interaction controlled by the REST OAuth service, the machine client has not control nor access to such resources but only waits until the flow is redirected through a receiving transition. Transitions may have guard conditions (`if`), perform internal tasks (e.g. `EvalAccessToken`, `Store`, etc), pass values or perform HTTP requests to external REST services (`POST`, `GET`, `Redirect`, etc.), corresponding to sender transactions ($\mathcal{T}_S$) in Decker's model. An input place (e.g. *S0*) can receive HTTP messages such as `POST`.

4.3 ReLL Based Dynamic Late Binding

The Petri Net is also modeled in XML as a simplified version of PNML (Figure 5). Concerns are separated in three layers: the REST service *resources* layer (which may introduce new resources when composing services (e.g. `http://ing.puc.cl/socialNetwork`); the ReLL service descriptions layer (which supports *dynamic late binding*); and the Petri Net model layer, that drives the client-server interaction.

Consider Figure 4, once the `socialNetwork` resource receives a `POST`, a token is generated and a `Fork` operation is performed separating the original token in two, one for each source service (LinkedIn and Facebook). Figure 5 details the token received at *s1*, it declares key-value pairs that were received in the POST request header, must be generated on run-time (e.g. timestamp), or are part of the machine-client internal state (e.g. cokies). Transition `Eval Access Token` evaluates whether the attribute `accessToken` (in this case, for LinkedIn) is set, and a guarded condition determines the next place. If the attribute is not set, tokens are moved to *s2* and the OAuth authentication process begins by triggering the `POST requestToken` transition.

This transition sends a message to LinkedIn's `requestToken` resource, the machine-client expects a response of type `requestToken`, `text/plain` (see Figure 2). The response is the input for the *s3* place and a new token (`LnToken`) will be minted by executing the regular expressions defined for `oauthUri` and `oauthToken` variables in the ReLL description. The URI for the `REDIRECT` transition will be minted by following the ReLL instructions for the `authorizeRq` link. To follow the minted URI, the machine-client must sent the `LnToken`, when performing a `get` operation on the `http` protocol, and expect a `text/plain` response; at most one link may be generated and the retrieved resource will correspond to the `authorize` type (Figure 2). The `authorize` resource represents the beginning of an out of band conversation between the service provider (e.g. LinkedIn) and a Web client controled by the provider. The machine-client `waits` until a response corresponding to the `callback` resource is received at place *s5*.

```
<place id="s1" resource="SocialNetwork">
  <token name="LnToken">
    <attr location="header" name="lnUser"/>
    <attr location="cookie" name="oauth_consumer_key" />
    <attr name="oauth_nonce" type="Random"/>
    <attr name="oauth_timestamp" type="Timestamp"/>
    <attr name="oauth_signature_method">HMAC-SHA1</attr>
    <attr location="cookie" name="oauth_callback" />
    <attr location="cookie" name="accessToken" />
    <attr name="oauth_version">1.0</attr>
    <attr location="cookie" name="connections" />
  </token>
</place>
...
<place id="s3" resource="requestToken">
  <token name="LnToken">
    <attr location="body" name="oauthUri"/>
    <attr location="body" name="oauthToken"/>
  </token>
</place>
<transition id="REDIRECT" link="authorize_rq">
  <token name="LnToken">
    <attr location="body" name="oauthUri"/>
    <attr location="body" name="oauthVerifier"/>
  </token>
</transition>
<place id="wait" resource="authorize"/>
<place id="s5" resource="callback">
...
```

Fig. 5. Petri Net XML snippet

An internal operation will store part of the received response (the `accessToken`), a new URI will be minted for the next transition (`GET connections`) and the token as detailed initialed for the `s1` place will be sent with updated values (cookies). The response of both LinkedIn and Facebook services will be merged and a new resource will be generated (i.e. `socialNetwork/1`).

A CPN tool was used to model and simulate the composition. An XML description for the Petri was manually created, and both Petri and ReLL descriptions were parsed, uploaded and executed by the machine-client. The machine-client is a refined version of RESTler [9]. Internally, the machine-client includes libraries for parsing and executing regular and XPath expressions, as well as an HTTP client to perform requests to REST resources and expose itself as a resource: the `SocialNetwork`. Transitions are fired according to an enablement component that handles the markings and tokens passed among places.

5 Conclusions

Service descriptions (e.g. ReLL) introduce coupling between clients and servers, however even loosely coupled services need a shared set of assumptions, and a more formal way of describing those assumptions will help service providers and consumers in service documentation and consumption, as evidenced by the currently existent REST APIs documentation. ReLL allows clients to detect whether some assumptions have changed (e.g. more links than expected are

served, the URIs have changed, the protocol have changed etc.), so that a proper action can be taken (e.g. extend the ReLL description and describe the new interface or even versioning the description).

By separating concerns in different layers (description, composition) instead of merging them in a full fledged service composition engine, the ReLL descriptions are reusable in scenarios other than the described in this paper (e.g. web crawling, and semantic integration), we expect that the Petri net descriptions could be also reused in complex scenarios where composition actually include composed resources. On the downside, ReLL descriptions and the composition itself are static. We plan to explore recent work on planning that make possible to introduce dynamic decisions and generate Petri Nets accordingly, as well as to explore the impact of explicit semantics in a dynamic composition scenario.

References

1. Overdick, H.: Towards resource-oriented BPEL. In: Pautasso, C., Gschwind, T. (eds.) WEWST. CEUR Workshop Proceedings, vol. 313, CEUR-WS.org (2007)
2. Alarcón, R., Wilde, E.: Linking data from restful services. In: Third Workshop on Linked Data on the Web, Raleigh, North Carolina (April 2010)
3. Pautasso, C.: Composing rESTful services with jOpera. In: Bergel, A., Fabry, J. (eds.) SC 2009. LNCS, vol. 5634, pp. 142–159. Springer, Heidelberg (2009)
4. Pautasso, C.: RESTful web service composition with BPEL for REST. Data Knowl. Eng. 68(9), 851–866 (2009)
5. Rosenberg, F., Curbera, F., Duftler, M.J., Khalaf, R.: Composing RESTful services and collaborative workflows: A lightweight approach. IEEE Internet Computing 12(5), 24–31 (2008)
6. Xu, X., Zhu, L., Liu, Y., Staples, M.: Resource-oriented architecture for business processes. In: APSEC, pp. 395–402. IEEE, Los Alamitos (2008)
7. Decker, G., Lüders, A., Overdick, H., Schlichting, K., Weske, M.: RESTful petri net execution. In: Bruni, R., Wolf, K. (eds.) WS-FM 2008. LNCS, vol. 5387, pp. 73–87. Springer, Heidelberg (2009)
8. Hadley, M.: Web application description language. World Wide Web Consortium, Member Submission SUBM-wadl-20090831 (August 2009)
9. Alarcón, R., Wilde, E.: Restler: Crawling restful services. In: Rappa, M., Jones, P., Freire, J., Chakrabarti, S. (eds.) 19th International World Wide Web Conference, pp. 1051–1052. ACM Press, Raleigh (2010)
10. Alarcon, R., Wilde, E.: From restful services to rdf: Connecting the web and the semantic web. School of Information, UC Berkeley, Berkeley, California, Tech. Rep. 2010-041 (June 2010)

Process Restructuring in the Presence of Message-Dependent Variables

Thomas S. Heinze[1], Wolfram Amme[1], and Simon Moser[2]

[1] Friedrich Schiller University of Jena,
07743 Jena, Germany
{T.Heinze,Wolfram.Amme}@uni-jena.de
[2] IBM Software Laboratory Böblingen,
71032 Böblingen, Germany
smoser@de.ibm.com

Abstract. When services interact, issues can be caused by service implementations being stateful because a stateful implementation requires a certain message exchange protocol to be followed. At present, a model of such a message exchange protocol is seldom complete and precise, mainly because the available analysis techniques for its derivation suffer from drawbacks: most prominently the neglect of data. Process restructuring allows for the increase of precision of such a data-unaware analysis by resolving conditional into unconditional control flow in service implementations and hence eliminating the need to consider data. But the restructuring approach so far has been restricted to cases where conditions of data-based choices have been defined over quasi-constant variables only. In this paper we introduce a restructuring technique that also allows us to resolve data-based choices with conditions over variables whose value is determined by the contents of incoming messages.

1 Introduction and Motivation

In *service choreographies* and *service orchestrations* issues can occur as soon as one service has a stateful implementation. This would be the case with services that are implemented as business processes, e.g. in languages like WS-BPEL [11] or BPMN [2]. Stateful services require another service that interacts with them to follow a certain protocol in order to prevent runtime errors, e.g. deadlocks. Consider an example: An auction house service offers an interface with two methods: (1) `Bid()` can be used to place a bid and (2) `Abort()` terminates the auction. It requires a bidding service interface for the auction participants with two methods: (1) `BidRequest()` - a notification that bids can be placed and (2) `BidClosure()` - a notification that the auction has finished. An auction participant service offers the bidding service interface and requires an auction house interface. Hence the services should be compatible since each service offers what the other requires. Considering the service implementation shown as BPMN in Fig. 1, it becomes obvious that an auction participant service must eventually invoke the `Abort()` method or the `Bid()` method (with an argument exceeding

E.M. Maximilien et al. (Eds): ICSOC 2010 Workshops, LNCS 6568, pp. 121–132, 2011.

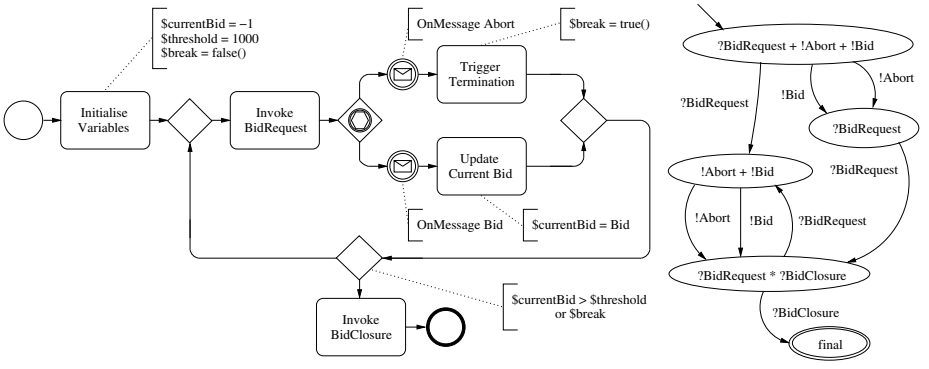


Fig. 1. Service implementation `AuctionHouseService` and its operating guideline

the bidding threshold), i.e. it must follow a certain *message exchange protocol* so that the service interaction terminates. Although there are many forms of this protocol, the underlying idea is the same: It is an automaton describing the externally visible behavior of the service. We will use the term *operating guideline*, coined by [7], for this automaton. Methods calculating an operating guideline analyze the control flow of a service implementation, mainly based on low-level Petri nets, and build the automaton [6,7]. On the right side of Fig. 1, the such derived operating guideline for the auction house service is shown. Therein, all paths are expected to conform to proper interactions with potential partner services. Edges correspond to exchanged messages from the perspective of a partner, i.e. an incoming message is denoted by an edge labeled `"?"` and an outgoing message by an edge labeled `"!"`. Furthermore, additional prerequisites for proper partner services are given as annotations of the automaton's states.

As indicated above, the operating guideline is derived by analyzing a low-level Petri net model of the service implementation's control flow. Data is overall neglected in this model in favor of a feasible analysis. However, by this means, conditional control flow must be mapped to *nondeterminism*, such that the outcome of a data-based choice is independent of its condition, i.e. whether the bidding cycle in the service implementation `AuctionHouseService` is entered or not is chosen arbitrarily within the Petri net model. Consequently, the control flow is over-approximated which hinders precise analysis and the derived operating guideline, shown in Fig. 1, thus provides only a coarse characterization for the behavior of the service. For instance, compatible partners, which send several bid messages `Bid` before awaiting the respective bid request messages `BidRequest`, are not represented and the exit of the bidding is also not explicitly depicted. In [5], we introduced a *process restructuring approach* which allows for the transformation of certain kinds of conditional into unconditional control flow. If effectually applied, there is no need to map the conditional control flow to nondeterminism in the Petri net model because it has been resolved and this source of imprecision can be avoided. As a result, the data-unaware derivation of operating guidelines is able to provide more precise results.

In this paper, we use this approach as the foundation for an advanced technique. Using the new technique then not only broadens the applicability of our restructuring approach, but also refines the derivation of operating guidelines by means of message contents. The remainder of the paper is structured as follows: Sect. 2 introduces the process model used. The restructuring technique is presented in Sect. 3, followed by a discussion of its application to our example in Sect. 4 and related work in Sect. 5. Finally, Sect. 6 concludes the paper.

2 Metamodel for Business Processes

A process model for our purposes must fulfill two requirements: capable of describing more than one business process language and able to precisely reflect the control flow and the data aspects of a service implementation. We therefore introduced *Extended Workflow Graphs* in [5]. Extended Workflow Graphs, abbreviated eWFG, are workflow graphs enriched by a notation of process data. Formally, a workflow graph is a directed graph $WFG = (N, E)$ such that $N = N_{Activity} \cup N_{DivGateway} \cup N_{ConvGateway}$ consists of

- $N_{Activity}$, i.e. nodes to represent activities,
- $N_{DivGateway}$, i.e. nodes to split the control flow in branchings or loops, or mark the beginning of a parallel section - a fork, a decision, or an IOR-split,
- $N_{ConvGateway}$, i.e. nodes to merge the control flow in branchings or loops, or mark the end of a parallel section - a join, a merge, or an IOR-join

and (1) there is exactly one start node $Start \in N$ and one stop node $End \in N$, (2) each diverging control flow node $n \in N_{DivGateway}$ has exactly one incoming edge and two or more outgoing edges, whereas each converging control flow node $n \in N_{ConvGateway}$ has exactly one outgoing edge and two or more incoming edges; each activity has exactly one incoming and exactly one outgoing edge, and (3) each node $n \in N$ is on a path from the start node to the stop node.

Mapping for a well-structured language like WS-BPEL to WFG is straightforward since the points where the control flow diverges and converges are well defined. For BPMN this is more complex because BPMN theoretically allows to model unsound [1] and unstructured processes. However, whether a process is sound can be checked with the methods from [9], and [10] even proposes a refactoring method to make an unstructured process structured. Once a WFG representation has been constructed, it has to be annotated with data flow information. In eWFGs, data is encoded by *Concurrent Static Single Assignment Form (CSSA-Form)*. The key advantage of CSSA-Form is that each variable is defined exactly once, which benefits process analysis. To guarantee this property, variables are renamed: Variable v becomes values $v_1, \ldots, v_n$, one for each of its definitions. As the property is considered static, the definition of a variable inside a control flow cycle is regarded as a single definition, although, the cycle may be executed more than once. Special handling is required if multiple definitions of a variable reach a node via different branches or threads, i.e. at nodes $N_{ConvGateway}$. In these cases, functions $\Phi(v_1, \ldots, v_n)$ are inserted to merge the

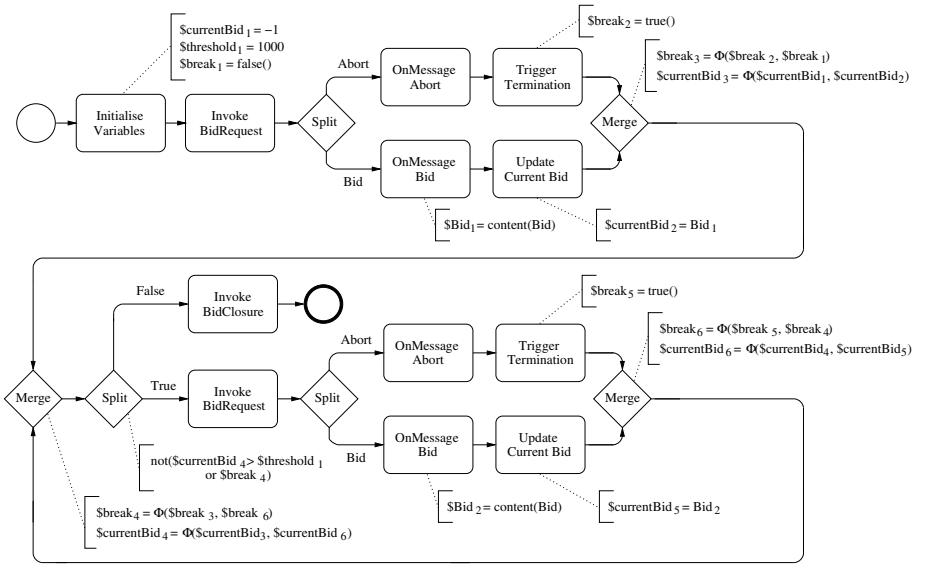


Fig. 2. Extended workflow graph of `AuctionHouseService` process from Fig. 1

confluent definitions $v_1, \ldots, v_n$ of a variable v. The value of such a *Φ-function* is one of its operands: v_i, if the i-th incoming edge denotes the branch taken at process runtime or the parallel thread whose associated operand is defined last.

The eWFG for our example is shown in Fig. 2. For each activity, a distinct node is created and interconnected according to the process's control flow. More complex control flow structures are mapped using pairs of nodes for diverging and converging control flow, i.e. `Split` and `Merge`. Thereby, the cycle in `AuctionHouseService`, which depicts a repeat-until loop, has been transformed into a while loop, due to technical reasons.[1] This can be done safely by extracting the first iteration of the loop and negating the loop condition. Process data are now represented by means of CSSA-Form, such that each static definition of a variable is assigned a unique name, e.g. $\$break_1, \ldots, \$break_6$ for variable `$break`. Furthermore, several *Φ*-functions have been introduced to merge confluent definitions of variables, e.g. $\$break_4 = \Phi(\$break_3, \$break_6)$.

3 Process Restructuring Technique

In [5], we presented a method for partially eliminating conditional control flow in business processes. More specifically, our *process restructuring approach* allows for the transformation of certain kinds of data-based choices such that data dependences become control dependences, while keeping the execution semantics of

[1] CSSA-Form relies on the presence of while loops. However, every structured loop can be transformed beforehand into a semantically equivalent while loop.

processes unchanged. As a result, conditions of the data-based choices are statically evaluable and can be replaced by unconditional control flow. To this end, the approach relies on a certain class of conditional control flow, characterized by conditions whose variables are defined over constants only. In our example in Fig. 1, the variable `$break` depicts such a *quasi-constant* variable, since it is only defined by the constants `false` and `true`. The condition of the data-based choice in the process is further based on a variable, i.e., `$currentBid`, which is defined by messages. In the following, we will describe a restructuring technique which can be also used in cases where condition variables are defined by messages.

3.1 Basic Restructuring Approach

The restructuring method in [5] has been tailored toward well-structured processes. This means it can cover WS-BPEL processes and a good subset of BPMN, but in its current form is not useable for unsound or unstructured processes. The method can also not be applied to data-based choices with data dependences crossing the boundaries of threads, that is, parallel sections of a process. Furthermore, we will focus on loops, i.e. well-structured control flow cycles, in the subsequent description, by the reason that the restructuring of a structured acyclic branching can be seen as special case of restructuring a loop.

A loop can be effectually processed by our method if the static value of any variable appearing in its loop condition correlates with particular paths in the control flow. Exploiting this property allows the transformation of the loop such that these implicit control dependences become explicit and can be used as substitutes of the loop condition. For instance, the value of a variable which is only defined by constants is in correlation with certain control flow paths, since any path determines the constant in effect. In [5], we called such variables *quasi-constant* and restricted our restructuring approach to loops whose conditions where defined over quasi-constant variables only. Separating the paths which are associated to different assignments of constants to condition variables enabled us to statically evaluate and remove loop conditions for this class of loops.

Restructuring is done in two steps: First, the loop is converted into a *loop normal form*. This normal form is characterized by the separation of all static control flow paths defining different values for variables of the loop condition. To obtain the normal form, nodes, excluding the loop header, where different definitions of condition variables converge, are resolved. After normalization, all definitions of condition variables only converge at the loop header node. Second, the remaining control flow paths which define differing values for condition variables, i.e. the dynamic paths coalesced at the loop header, are separated. The loop is divided into copies of its body, called *loop instances*. Each instance represents the execution of the loop for a certain assignment of condition variables. For loops with conditions of quasi-constant variables, the assignments consist of constants only. Thus, the loop condition is evaluable for each instance and can be replaced by unconditional control flow. Since the number of possible assignments, and so the number of loop instances, is bound by the amount of constants assigned to variables, the transformation is guaranteed to terminate.

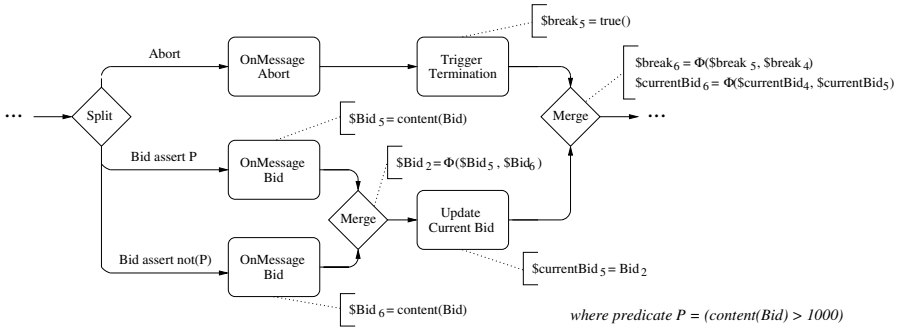


Fig. 3. Splitting incoming message activity `OnMessage Bid`

3.2 Message-Dependent Variables

We now enlarge the applicability of our restructuring approach to cases where a loop condition contains, besides quasi-constant variables, variables that are defined by messages. We call variables of this kind *message-dependent*. To restructure loops with conditions over message-dependent variables, we need a way to associate the possible values of such variables to the control flow.

Depicting messages not only as events, but rather distinguishing the contents of messages, allows for a more fine-grained representation of message-dependent variables. We refine the representation of a message-dependent variable by help of *assertions* for its defining messages. The assertions are based on propositions over predicates, which characterize the possible contents of messages. The definition of a variable by an incoming message is then split into multiple definitions, one for each assertion. This *predicate-based abstraction* allows us to link the values of message-dependent variables as distinct variable definitions to the control flow.

Since our goal is to eliminate conditional control flow of loops, the assertions for incoming messages are derived from the respective loop condition. To this end, the basic predicates of the loop condition are analyzed. Identified predicates can be categorized into quasi-constant predicates, i.e. predicates where only quasi-constant variables are used, and message-dependent predicates, i.e. predicates where additionally message-dependent variables are used. For each message-dependent predicate P, the predicate P and its negation $\neg P$ are used as assertions for the respective incoming message. If a message-dependent variable is used in more than one predicate, all propositions over these predicates form assertions for the message: $P \wedge Q$, $\neg P \wedge Q$, $P \wedge \neg Q$, $\neg P \wedge \neg Q$ for predicates P and Q.[2] The such derived assertions are then utilized for splitting the definitions of those message-dependent variables that are used in the loop condition. Note that, in case of two or more message-dependent variables used in a single predicate, this approach only provides a conservative approximation since we do not examine the actual logical functions realized by condition predicates.

[2] If predicates are logically correlated, e.g. $P = (X > 5)$ and $Q = (X < 10)$, certain assertions are contradictory and can be therefore omitted, e.g. $\neg P \wedge \neg Q$.

Fig. 3 shows the result of splitting one of the incoming message activities contained in the eWFG shown in Fig. 2. The activity is split into two distinct activities, which are separated using message events based on complementary assertions for the incoming message, i.e. *assert* P and *assert not*(P). The assertions are grounded on predicate P, which corresponds to expression $currentBid_4 > threshold_1$ of the loop condition. Therein, referenced variables have been updated according to their actual values, i.e. $threshold_1$ has been replaced by constant 1000 and $currentBid_4$ by the contents of message `Bid`.

3.3 Extended Process Restructuring

A loop with condition over *quasi-constant* and *message-dependent* variables, which respects the restrictions listed in Sect. 3.1, can be restructured in an automated fashion using three steps. First, data dependences for the loop condition are identified. Utilizing the derived information, the definition of each message-dependent condition variable is split into multiple definitions, as described above. Next, the loop is transformed into its *loop normal form* by employing the normalization algorithm in [5]. Finally, the loop is divided into its *loop instances* by using the adaption of the instantiation algorithm introduced below.

In the instantiation process, loop instances are iteratively generated and used as loop replacement. Each instance is guarded by a copy of the data-based choice containing the loop condition, called *instance guard*. If the condition can be evaluated, the guard is replaced by an edge to the exit node of the loop, when evaluation yields false, or, when evaluation yields true by an edge to the instance. The instance itself points to subsequent guards, which are processed in the following iterations. In order to assure the static evaluation of guards, a new notion of loop instance is used. So far, an instance has been considered an execution of the loop with respect to an assignment of constants to variables. When also allowing condition variables defined by messages, this concept is not enough. Instead, we now consider an instance to be an execution of the loop body with respect to an arbitrary *assertion* for the state of condition variables. The state of a quasi-constant variable is then described as value assignment, depicting the constant in effect. The state of a message-dependent variable is characterized by exploiting the introduced assertions for incoming messages. Since these assertions were chosen so that they matched the predicates of the loop condition, the condition can be evaluated for each instance guard by help of elementary logical reasoning and constant expression evaluation. Due to the finite number of predicates, condition variables and assigned constants, the number of loop instances is bound and the instantiation algorithm is again guaranteed to terminate.

The adapted instantiation algorithm is shown in Fig. 4. Procedure *instantiate* consecutively processes instance guards, i.e. data-based choices containing the loop condition, until no further guard exist. To evaluate the condition of a guard, a new assertion A is generated based on the assignment of variables valid at this guard. A constant assignment is thereby added to A for quasi-constant condition variables and assertions of the incoming messages are taken over for message-dependent condition variables. Evaluating the guard based on assertion A allows

```
procedure instantiate(eWFG = (N, E)) is
    while (∃ guard ∈ N such that guard is instance guard) do
        let values be the assignment of condition variables valid at guard;
        A = create empty assertion;
        foreach single assignment v_{c_i} ← v_i in values do
            if (v_i is a message-dependent variable) then
                let M be the incoming message defining v_i;
                let A_M be the assertion valid for message M;
                add ((v_{c_i} == content(M)) ∧ A_M) to assertion A;
            else add (v_{c_i} == v_i) to assertion A;
            end if;
        end for;
        if (evaluate(condition(guard), A) == true) then
            let Instance_A be the loop instance associated to assertion A;
            if (not(Instance_A ⊆ eWFG)) then
                eWFG = eWFG ∪ Instance_A;
            end if;
            let entry_{Instance_A} be the entry node of Instance_A;
            E = (E \ {(predecessor(guard), guard)})
                    ∪ {(predecessor(guard), entry_{Instance_A})};
        else E = (E \ {(predecessor(guard), guard)})
                    ∪ {(predecessor(guard), exit)};
        end if;
    end while;
end.
```

Fig. 4. Adapted algorithm for loop instantiation

us to replace it by an edge to the loop exit, when evaluation yields false, or by an edge to the loop instance $Instance_A$ associated to assertion A, otherwise. If no such instance yet exists, a new one is created. The restructured workflow graph for our example is shown in Fig. 5. Only a single instance has been created while instantiating the loop of the example since only the guard of the instance with assertion $(break_4 == false \wedge currentBid_4 == content(Bid) \wedge not\, content(Bid) > 1000) \wedge threshold_1 == 1000)$ resulted true. Obviously, the data-based choice of the eWFG shown in Fig. 2 was replaced by unconditional control flow.

4 Application to Business Process Analysis

In this section we present the application of our process restructuring technique to the derivation of *operating guidelines*. Existing methods for the construction of operating guidelines are based on a low-level Petri net model of business processes [6,7]. Despite being conservative, such an abstraction provokes imprecise analysis results. In the following, we will show how our process restructuring technique is able to remedy this kind of imprecision for data-based choices with conditions of *message-dependent* and *quasi-constant* variables.

The `AuctionHouseService` example from Fig. 1 is an instance of a process whose derived operating guideline was incomplete. It is to be expected that the

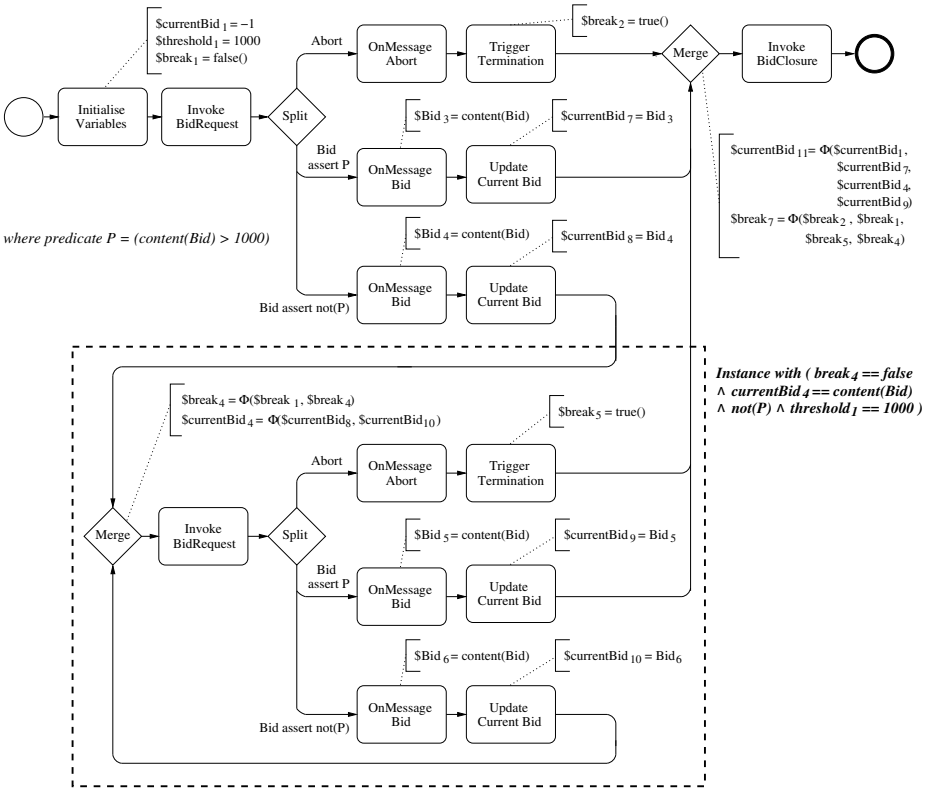


Fig. 5. Restructured workflow graph of `AuctionHouseService`

operating guideline derived from the restructured eWFG will include the missing partner processes. As mentioned, derivation of operating guidelines requires a *Petri net model*. The restructured eWFG of `AuctionHouseService` hence must be mapped onto that model. Since a similar translation from WFG into Petri nets has already been defined in [1], a slightly adopted mapping can be used in our case. Analyzing the Petri net derived from the restructured eWFG results in the operating guideline shown in Fig. 6.[3] Compared to the conventional operating guideline in Fig. 1, the missing compatible partner processes are now represented. For example, the path which traverses edges `!Bid[assert not P]`, `!Bid[assert not P]`, `?BidRequest`, `?BidRequest`, `?BidRequest`, `!Bid[assert P]`, `?BidClosure` depicts a partner that sends two bids not exceeding the threshold value, receives the associated plus an additional bid request, submits a bid exceeding the threshold value, and finally picks up the bid closure message.

As can be furthermore seen, the new operating guideline explicitly models the exit condition of the process. A partner can either send `Abort` or a bid exceeding

[3] Derivation of operating guidelines is only possible for bounded communication channels [7]. Therefore, we depict the operating guideline for channels of size $k = 3$.

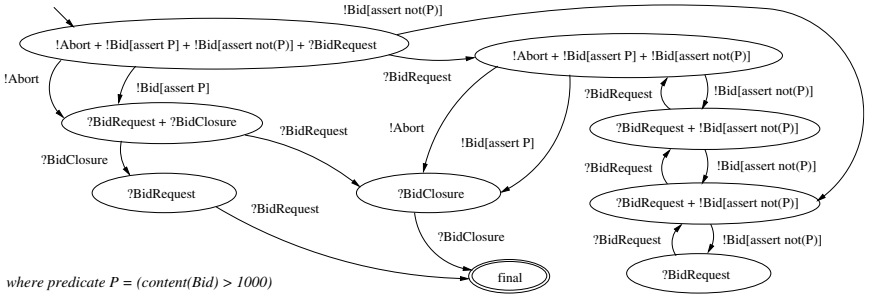


Fig. 6. Refined operating guideline of `AuctionHouseService`

the threshold value, i.e. `Bid[assert P]`, to reach the final state. In comparison, the operating guideline in Fig. 1 masks this exit condition. A partner is required here to determine whether the bidding is over or not by the receipt of message `BidClosure` or `BidRequest`. Masking the exit condition causes the conventional operating guideline to contain spurious behavior, which is not realized by the process. Thus, a path is included where, albeit `Abort` has been transmitted, further bid requests can be received and bids sent, which is not possible with process `AuctionHouseService`. On the one hand, this spurious behavior does not induce a deadlock, but, on the other hand, might result in a livelock. In contrast, the new operating guideline does not contain spurious behavior.

Generally speaking, the operating guideline in Figure 6 is a refinement of the conventional operating guideline such that the therein hidden data-based choice of process `AuctionHouseService`, defining its exit condition, is now made explicit. This refinement is possible since the data-based choice is based on a condition of *message-dependent* and *quasi-constant* variables and our *process restructuring technique* can be effectually applied. As a result, we are able to transform conditional into unconditional control flow and to provide a precise model of the control flow without the need of considering data dependences, which benefits the subsequent Petri-net-based derivation of operating guidelines.

However, there is a price: The potentially confidential threshold value used in the process needs to be announced, as predicate $P = (content(Bid) > 1000)$. Since the conventional operating guideline already provides a description of some compatible partners, it may be in question whether to use the refined operating guideline. This is not an option if an empty operating guideline has been initially derived. Consider the variation of `AuctionHouseService` shown in Fig. 7. Therein, no bid request is sent, like in Fig. 1, but each bid is acknowledged by a confirmation message. Consequently, the internal choice whether to receive further bids or to exit the bidding is not signaled. The analysis in [7] regards this process as not usable and derives an empty operating guideline, which is obviously wrong. For instance, the interaction with a partner which sends a single bid exceeding the threshold value receives the confirmation and the bid closure message does not deadlock. To derive a non-empty operating guideline, a method which allows for a more precise analysis of the process's conditional control flow

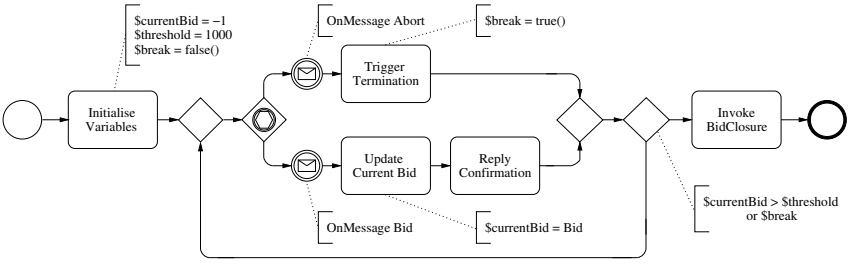


Fig. 7. Variation of `AuctionHouseService`

is mandatory. Moreover, the operating guideline must announce the threshold value used in the branching condition of the process. Thus, describing process behavior by means of a *message exchange protocol* is required here to take data into account, i.e. the contents of exchanged messages.

5 Related Work

Most approaches for control-flow analysis of business processes neglect data, particularly when utilizing Petri nets. If precise analysis of a process's control flow is of concern, a common argument is to use *high-level Petri nets*, which allow for a data-aware process model. In this regard, the inclusion of message contents is sketched in [6] by means of unfolding a high-level into a low-level net, which can be used as analysis input. However, this approach is only feasible for a finite data domain. By help of *predicate abstraction* [4], an infinite domain can be reduced to a finite domain. But, the thereto required predicates need to be properly derived, which is especially challenging in the presence of loops [3]. In our example, predicate `$currentBid` > `$threshold` of the loop condition is apparently a good candidate. The difficulty is to identify and adequately incorporate the data dependences of used variables, i.e. the fact that `$threshold` denotes a constant and `$currentBid` the contents of messages.

In [8], a Petri net model of business processes is extended with a notion of process data, based on an abstract and finite data model. As a result, the such extended process model can also be checked for properties like soundness [1]. However, in contrast to our approach, the used process model is conceptual, i.e. underdefined, and the utilized data abstraction must be provided manually.

6 Conclusion and Outlook

In this paper, we have presented a process restructuring technique and its benefits for the analysis of business processes. The technique allows us to resolve data-based choices with conditions over message-dependent and quasi-constant variables. As a result, data dependences of resolved choices do not need to be taken into account in a subsequent, potentially data-unaware, control-flow

analysis. In this way, we were able to demonstrate a refined derivation of message exchange protocols for business processes by means of message contents.

The main issue for future work is the thorough validation and evaluation of our restructuring technique. Having laid its methodological foundation within this paper, we want to assess the practical use of the presented approach by applying it to a collection of real-world processes. Unfortunately, we are not aware of a representative set of business processes, i.e. a common benchmark, which can be used for that purpose. In consequence, we will need to assemble a comprehensive collection of realistic business processes first.

References

1. van der Aalst, W.M.P., Hirnschall, A., Verbeek, H.M.W.: An Alternative Way to Analyze Workflow Graphs. In: Pidduck, A.B., Mylopoulos, J., Woo, C.C., Ozsu, M.T. (eds.) CAiSE 2002. LNCS, vol. 2348, pp. 535–552. Springer, Heidelberg (2002)
2. Business Process Model and Notation (BPMN) Version 2.0. OMG Standard, Object Management Group/Business Process Management Initiative (2009)
3. Flanagan, C., Qadeer, S.: Predicate Abstraction for Software Verification. ACM SIGPLAN Notices 37(1), 191–202 (2002)
4. Graf, S., Saidi, H.: Construction of Abstract State Graphs with PVS. In: Grumberg, O. (ed.) CAV 1997. LNCS, vol. 1254, pp. 72–83. Springer, Heidelberg (1997)
5. Heinze, T.S., Amme, W., Moser, S.: A Restructuring Method for WS-BPEL Business Processes Based on Extended Workflow Graphs. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) BPM 2009. LNCS, vol. 5701, pp. 211–228. Springer, Heidelberg (2009)
6. Lohmann, N.: A Feature-Complete Petri Net Semantics for WS-BPEL 2.0. In: Dumas, M., Heckel, R. (eds.) WS-FM 2007. LNCS, vol. 4937, pp. 77–91. Springer, Heidelberg (2008)
7. Lohmann, N., Massuthe, P., Wolf, K.: Operating Guidelines for Finite-State Services. In: Kleijn, J., Yakovlev, A. (eds.) ICATPN 2007. LNCS, vol. 4546, pp. 321–341. Springer, Heidelberg (2007)
8. Sidorova, N., Stahl, C., Trčka, N.: Workflow Soundness Revisited: Checking Correctness in the Presence of Data While Staying Conceptual. In: Pernici, B. (ed.) CAiSE 2010. LNCS, vol. 6051, pp. 530–544. Springer, Heidelberg (2010)
9. Vanhatalo, J., Völzer, H., Koehler, J.: The refined process structure tree. Data & Knowledge Engineering 68(9), 793–818 (2009)
10. Vanhatalo, J., Völzer, H., Leymann, F., Moser, S.: Automatic Workflow Graph Refactoring and Completion. In: Bouguettaya, A., Krueger, I., Margaria, T. (eds.) ICSOC 2008. LNCS, vol. 5364, pp. 100–115. Springer, Heidelberg (2008)
11. Web Services Business Process Execution Language Version 2.0. OASIS Standard, Organization for the Advancement of Structured Information Standards (2007)

Simple Metric for Assessing Quality of Service Design

George Feuerlicht[1,2]

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

Abstract. Service design has been the subject of intense research interest and there is a wide agreement about the key principles that lead to good quality design of services. However, there is evidence that achieving good quality design of services in practice is difficult and that many service oriented applications suffer from low levels of reuse and are difficult to evolve. Recent research efforts include attempts to develop reliable metrics for assessing design quality of service oriented applications. In this paper we argue that poor reuse of services can be largely attributed to coarse-granularity document-centric services that are used extensively by SOA practitioners. We briefly discuss reusability in the context of domain-specific service-oriented applications and propose a simple design metric that estimates the level of data coupling between services based on orthogonality of interface data structures.

Keywords: Service Design Metrics, Service Coupling, Service Cohesion, SOA.

1 Introduction

Notwithstanding extensive research efforts service design still present a number of important challenges, in particular with regards to service reuse. Software reuse has been studied extensively in the context of object-oriented programming and more recently Service-Oriented Architecture (SOA). Most experts consider reuse as essential to reduce the costs associated with design, development, testing and maintenance of software [1], [2], however it is also widely recognized that achieving reuse in practice involves considerable design-time effort [3].

The promise of software reuse was one of the most important motivations for the introduction of the object-oriented approach to software construction, but there is evidence that this promise has not been fully realized [3, 4]. Likewise, component-based development was regarded as a solution to poor levels of software reuse, but lack of standards for component interoperability prevented significant improvements in software reuse [5, 6]. SOA is the latest approach that is widely regarded as having the potential to significantly improve software reuse and avoid duplication of applica-

E.M. Maximilien et al. (Eds): ICSOC 2010 Workshops, LNCS 6568, pp. 133–143, 2011.

tion functionality [7]. However, the notion of reuse in context of services has a number of interpretations and often refers to *wrapping* legacy application components as Web Services, rather than reuse as a result of design-time effort. Software reuse is essentially a programming concept popularized with the advent of object-oriented programming and applied to program modules (i.e. methods) and software components [8]. Reuse involves the invocation of a method in multiple application contexts via a well-defined interface that encapsulates data structures. Reuse is achieved by designing classes and methods to be self-contained and highly cohesive with inheritance and polymorphism playing a key role in ensuring that a single method can be reused in different applications. This mechanism (i.e. inheritance and polymorphism) is not compatible with the SOA paradigm and service reuse relies entirely on designing *composable* services, i.e. services that can be reused in multiple service compositions. This requires careful design-time considerations that lead to minimization of coupling and maximization of service cohesion, resulting in orthogonal services avoiding duplication of service functionality and associated data structures.

Achieving good levels of reuse can be particularly challenging as SOA is often associated with interactions based on coarse-grained document-centric services that limit the potential for reuse. This is particularly true in domain-wide situations (e.g. travel domain) where standardizations of messages by industry consortia often results in complex message structures that incorporate the requirements of a wide range of user organizations. Interactions between parties in such environments are typically achieved using Web Services based on a standard industry-wide XML document specification with individual documents constituting message payloads and SOAP providing the transport mechanism (alternatively, e-business interactions are implemented using REST services [9]). Although regarded as service-oriented applications, the design of such domain-wide document-centric applications does not normally follow established service design principles (see section 2.1 for discussion of service design principles) as it is constrained by message structures based on an existing domain specification, e.g. OTA (Open Travel Alliance) message specification standard [10].

In document-centric applications, documents constitute a key artifact of business communications, and design of standard documents (i.e. message structures) is typically based on Document Engineering [11-13] or similar methods that construct documents by identifying and aggregating common data elements (e.g. ebXML Core Components [14]). Using Core Components ensures uniform structure and semantics of data elements across the entire specification, but at the same time embedding Core Components into multiple business documents causes high levels of data coupling with corresponding impact on applications when the ebXML specification evolves. Core Components effectively become extensions of the type system of the underlying data model and are impossible to change without significant impact on existing domain applications. Embedding common data elements in business documents is promoted in Document Engineering literature as a technique for achieving reuse, but this approach differs fundamentally from software reuse as it applies to programming environments. It can be argued that the externalization of data structures implicit in

the document-centric approach limits, rather than enhances, reuse [15]. Similar situation arises in the OTA specification where complex data structures are embedded in multiple OTA messages. For example, the TravelPreferences complex element type – an XML data structure consisting of 11 simple elements with 2 levels of nesting is embedded in 2 parent complex element types (AirSearch and OriginDestinationInformation) and 3 message structures (OTA_AirAvailRQ, OTA_AirFareDisplayRQ, and OTA_AirLowFareSearchRQ). Changes in the TravelPreferences element type as the specification evolves will affect all parent data types and messages, with corresponding impact on existing applications.

Recent research efforts focus on developing reliable metrics for assessing design quality of service oriented applications, and a number of such metrics have been proposed and tested. However, in general these metrics are not suitable for assessing design of domain-wide document-centric applications. In this paper we focus on design of domain-wide services and propose a metric that indirectly estimates the level of service orthogonality based on data coupling between service interfaces. We first review service design principles and recent literature dealing with service design metrics (section 2). In the following section (section 3) we describe our proposal for a simple DCI (Data Coupling Index) metric that estimates the level of data coupling between services and provides a measure of the quality of service design based on orthogonality of interface data structures. We illustrate the application of the metric using an example based on the Open Travel Alliance specification (section 4). We conclude by noting that further research is needed to evaluate the sensitivity of the DCI metric to quality of design by comparing different design strategies for the same application scenario. We also plan a comparative study of DCI with other service design metrics (section 5).

2 Related Work

In this section we firstly review literature dealing with service design principles (section 2.1) and then review recent work on formulating suitable metrics for accessing the design of services (section 2.2).

2.1 Service Design Principles

Design plays a key role in improving service reuse and there is a wide agreement about design principles that include maximization of cohesion and minimization of coupling [16, 17]. Coupling refers to the degree of interdependence between services, and has been used in traditional software design as an indicator of software quality [18]. The concept of coupling has many interpretations; the term *loose coupling* has been used in the literature to signify independence from the underlying technology platform, stateless message-oriented interactions, asynchronous operation, and also has a number of other interpretations [19, 20]. Coupling as it applies to module design has been extensively explored and a number of different types of coupling, including *data coupling*, *control coupling*, and *stamp coupling* identified. It is generally recognized that while

limited level of coupling is essential (i.e. in order for software modules to interact they must share common interface parameters), excessive coupling is undesirable as it makes application systems *brittle* and limits the ability to adapt to change [20]. To understand the impact of coupling on reusability of services we need to differentiate between data coupling and stamp coupling [19, 21]. Data coupling refers to a situation where coupling takes place via simple data parameters that constitute the interface of a software module (e.g. service operation) and can be minimized by limiting the interface data parameters to those that are necessary to implement the service operation [22]. Stamp coupling refers to a situation where data is exchanged between services in the form of composite data structures (e.g. XML structures consisting of many elements and multiple levels of nesting) as is the case in document-centric services that use complex XML message payloads. Such coarse-grained services externalize complex data structures but tend to use only a part of the data structure to perform a given business function, causing unnecessary coupling. From a software engineering viewpoint, stamp coupling is undesirable as it violates the principle of data encapsulation and inhibits service evolution. Avoiding stamp coupling increases reusability by creating services with simpler interfaces and greater reuse potential.

Cohesion refers to the strength of the relationship between elements within a software module [23]. Maximizing service cohesion results in low coupling and minimization of the impact of changes on related services improving the stability of service-oriented applications and increasing potential for reuse. The highest level of cohesion is *functional cohesion* that refers to a situation where all the service operations are highly interrelated and contribute to the execution of a single clearly defined task [16]. Positive impact of functional cohesion on software reuse is documented in the literature [23, 24].

Service granularity (i.e. the scope of functionality of a service) is a key determinant of reuse, therefore careful consideration must be given to service design to establish an optimal level of granularity for a given set of application requirements [22]. There is a close relationship between coupling, cohesion, granularity, and reuse [25]. In general, highly cohesive services tend to lead to orthogonal design avoiding functionality overlap, and at the same time reducing coupling and granularity, and enhancing reuse. The design choice is typically a compromise between fine-grained services that implement *atomic* operations and exchange limited amount of data, and coarse-grained services that implement high-level business functions. Coarse-grained, document-centric design is used extensively in industry-wide message standards such as the OTA specification. The use of complex data structures as messages payloads results in stamp coupling, increasing the interdependencies between services and reducing reuse potential. The use of fine-grained services, on the other hand, may result in complex interaction dialogues and a complicated failure recovery [17]. Studies have confirmed that fine-grained services produce loosely coupled service-oriented applications with good maintenance properties [26] and methodological frameworks have been proposed to support making decisions about the optimal level of service granularity for a given implementation scenario [25, 27]. However, practitioners often make decisions about the level of service granularity using performance based heuristics, rather than a methodological framework.

2.2 Service Design Metrics

While the underlying principles of service design are extensively documented in the literature, little work has been done so far on how the adherence to these principles may be quantitatively measured [28]. Metrics for evaluating the quality of service design are necessary to allow the comparison of different design strategies and evaluation of their impact on service reuse. Several authors have proposed service design metrics that are based on metrics originally developed for structured programming and object-oriented systems. Chidamber et al. proposed the Lack of Cohesion in Methods (LCOM) metric that evaluates similarity of methods for a given class by counting the number of method pairs whose similarity is zero minus the number of method pairs whose similarity is not zero, where similarity of methods is defined as the intersection of the sets of instance variable used by the methods [29]. This metric has been used as the basis for developing metrics for service oriented systems by other authors. For example, the Service Interface Data Cohesion metric (SIDC) proposed by Perepletchikov et al. [30] measures cohesion by comparing the messages of service operations based on data types. Sindhgatta et al. developed a comprehensive set of metrics to measure service cohesion, coupling, and composability and applied these metrics to case studies in order to evaluate their applicability to practical scenarios [28]. Two variants of the LCOM metric ($LCOS_1$, $LCOS_2$) for use with services were adapted, and several additional metrics have been proposed by the authors in order to evaluate service and message coupling. These metrics include: Service Operational Coupling Index (SOCI) - a measure of dependence of a service on the operations of other services, Inter-Service Coupling Index (ISCI) – based on the number of services invoked by a given service, and Service Message Coupling Index (SMCI) that measures the dependence of a service on the messages derived from the information model of the domain (i.e. messages that service operations receive as inputs, and produce as output via the declared interface). Finally, Sindhgatta et. al propose several metrics dealing directly with service reuse and composability, including Service Reuse Index (SRI), based on the number of existing consumers of a service, Operation Reuse Index (ORI) that counts the number of consumers of a given operation, and Service Composability Index (SCOMP) defined on the basis of the number of compositions in which the service is a (composition) participant and the number of distinct composition participants which succeed or precede the service. Service granularity is closely related to reuse and composability and is evaluated using Service Capability Granularity (SCG) and Service Data Granularity (SDG) metrics, where higher values indicate coarser granularity (i.e. larger functional scope).

3 Proposed Service Design Metric

Most of the above described metrics assume that the quality of service design can be assessed using similar measures to those used for object-oriented applications. In general, the metrics assume a service model that consists of a set of services $S= \{s_1, s_2 \ldots s_S\}$ with operations $O(s) = \{o_1, o_2, \ldots o_O\}$ and interfaces formed by input and output messages M(o) [31]. This assumption is difficult to support in document-centric

services used in domain-wide e-business applications, as these types of services do not normally involve service operations; services are implemented using the request/response message exchange pattern that delivers standard (XML) documents within a SOAP envelope. Furthermore, such document-centric services typically implement high-level business functions and do not exhibit inter-service coupling (i.e. services do not *call* other services), making metrics based on the number of service invocations (e.g. ISCI) unsuitable.

Our proposal is to measure the quality of service design based on orthogonality of services. Lack of orthogonality of a set of services involves duplication of functionality and data structures and is associated with high levels of coupling and low levels of service cohesion, reducing the potential for reuse. To maximize orthogonality, each service (or service operation) should implement a single distinct atomic function; however this is not a practical solution in most application scenarios making some level of coupling is inevitable.

Orthogonality of services is difficult to measure directly and we rely on an indirect measure that estimates the level of data coupling between services using a Data Coupling Index (DCI). High values of DCI indicate low orthogonality of service design with corresponding negative impact on reuse and adaptability.

In document-centric SOA environments (e.g. as represented by OTA travel services) services are typically implemented using the request/response message exchange pattern. For a service S, the interface contract is given by:

$$S(M\_RQ, M\_RS)$$

where M_RQ and M_RS are the request and response messages, respectively

The request and response messages are aggregations of schema elements $\{se_1, se_2, se_3, \ldots.se_n\}$ that constitute the underlying data model. Although this is often not explicitly recognized, domain-wide applications are associated with an underlying data model from which data elements are drawn to form the various messages. Structure of the messages, in particular the extent of their orthogonality is critical in ensuring adaptability of the design.

We now define the Data Coupling Index (DCI) that is computed as the sum of the number of shared schema elements for each interface message pair combination; more specifically DCI is the cardinality of the intersection of schema element sets for the corresponding interface messages divided by the number of message pair relationships r:

$$DCI = \frac{1}{2r}\sum_{j,k=1}^{N} |M_j \cap M_k| \text{ ; where } j \neq k, \text{ and } r = \sum_{i=1}^{N-1} i$$

Where service interface is formed by the union of schema elements that participate in request and response messages and is denoted by M.

4 Evaluation of DCI Metric Using OTA Air Messages

We now proceed to evaluate the above defined metric using a set of OTA Airline (Air) messages. We limit our example to a subset of 12 air services shown in Table1.

Table 1. OpenTravel Air Messages

Ident.	OpenTravel Message	Business Functionality
AIR01	OTA_AirAvailRQ/RS	Search & Availability
AIR02	OTA_AirBookRQ/RS	Reservation Management: Booking
AIR03	OTA_AirBookModifyRQ	Reservation Management: Modification
AIR04	OTA_AirCheckInRQ/RS	Passenger Check-in & Check-out
AIR05	OTA_AirDemandTicketRQ/RS	Ticket Fulfillment
AIR06	OTA_AirDetailsRQ/RS	Descriptive Information: Flight leg and Codeshare
AIR07	OTA_AirFareDisplayRQ/RS	Fare Search & Display (No Availability)
AIR08	OTA_AirFlifoRQ/RS	Descriptive Information: Flight Operation
AIR09	OTA_AirPriceRQ/RS	Fare Pricing
AIR10	OTA_AirRulesRQ/RS	Fares Rules: Fare Basis & Negotiated Fares
AIR11	OTA_AirScheduleRQ/RS	Descriptive Information: Flight Schedules
AIR12	OTA_AirSeatMapRQ/RS	Seat Availability & Information

The OTA Air messages implement various business functions related to airline travel, such as checking flight availability, flight booking, etc. Consider, for example the Search and Availability of flights business function implemented as a service (e.g. Web Service) with the interface:

AIR_AVAILABILITY(Air_AvailabilityRQ, Air_AvailabilityRS)

Where Air_AvailabilityRQ and Air_AvailabilityRS are the request and response messages specified by Open Travel Alliance in the form of XML schema. OpenTravel Alliance XML Schema messages are widely adopted by companies that implement travel applications (e.g. Sabre [32]) and follow strict naming conventions and design guidelines [33]. Similar to ebXML, OTA defines common data types that form a "repository of reusable XML Schema components" and are used to construct the various messages. OTA differentiates between *complex types* (types that contain multiple data elements) and *simple types* (types that contain a single data element). The Air messages included in our example (AIR01-AIR12) contain 44 complex element types that expand into 649 simple data elements. The calculation of DCI is based on complex elements, i.e. the cardinality of the intersection of the schema elements in interface message pairs shown in Table 2 are counts of common complex elements shared by a given interface pair. Only the top level complex elements are taken into account, i.e. coupling at lower levels of nesting is ignored in the calculation of DCI. For example, service interfaces A01 (OTA_AirAvailRQ/RS) and A02 (OTA_AirBookRQ/RS) share 4 top level common complex elements.

Using the values in Table 2, we can compute:

$$r = 66 \text{ and } DCI = 3.56$$

DCI value (3.56) expresses the average number of common complex elements per interface message pair, i.e. on average OTA Air interface messages share 3.56 top level complex elements. The number of shared interface elements depends on the size of the interfaces under consideration, i.e. interfaces that include a larger number of complex elements are more likely to share more complex elements with other

Table 2. Number of common complex elements for interface message pairs

	A01	A02	A03	A04	A05	A06	A07	A08	A09	A10	A11	A12
A01	**x**	4	3	3	3	3	5	4	4	3	6	3
A02	4	**x**	11	4	4	3	4	3	6	3	4	4
A03	3	11	**x**	4	4	3	3	3	6	3	3	4
A04	3	4	4	**x**	5	3	3	3	4	3	4	4
A05	3	4	4	5	**x**	3	3	3	4	3	3	4
A06	3	3	3	3	3	**x**	2	3	3	3	3	3
A07	5	4	3	3	3	2	**x**	2	3	3	3	3
A08	4	3	3	3	3	3	2	**x**	3	3	4	3
A09	4	6	6	4	4	3	3	3	**x**	3	3	4
A10	3	3	3	3	3	3	3	3	3	**x**	3	3
A11	6	4	3	4	3	3	3	4	3	3	**x**	3
A12	3	4	4	4	4	3	3	3	4	3	3	**x**

interfaces. A more meaningful measure is the ratio of DCI over ANCT (Average Number of Complex Types per interface) - a *normalized* DCI index:

$$\text{NDCI} = 3.56/11.67 = 0.30$$

The interpretation of NDCI is that on average 30% of top level complex elements are shared with other interfaces. This value is probably excessive and indicates a high level of stamp coupling between OTA Air interface messages.

5 Conclusions and Further Work

While there is wide agreement about the key principles that lead to good quality of service design, experience with SOA projects indicates that achieving reuse and adaptability of services is difficult in practice. We have argued in this paper and elsewhere [34] that extensive use of coarse-grained document-centric services inhibits reuse and results in suboptimal design.

We have briefly reviewed existing service design metrics and identified their limitations in the context of domain-wide applications that are characterized by complex and overlapping message structures that incorporate the requirements of a wide range of user organizations. We have proposed a simple metric - Data Coupling Index (DCI) that estimates the level of data coupling between services and provides a measure of the quality of service design based on orthogonality of interface data structures.

Our initial study includes calculation of DCI using a set of OTA Air messages. While some observation about the quality of design can be made using DCI (or the normalized version NDCI), the metric needs to be validated using a larger set of messages and for different design strategies. Evaluating the DCI metric for different design strategies for a given application scenario and comparing the results for services with varying levels of granularity will validate its practical applicability. In calculating the DCI metric we restricted our analysis to top level complex data elements, i.e. without taking into consideration complex elements that appear at lower levels of the

hierarchy when the message structure is expanded further. This simplification avoids having to compare large number of data elements for each message pair, but can produce a biased result as some complex elements are deeply nested and include a large number of simple elements (e.g. PriceInfo element consists of 54 simple elements with 7 levels of nesting), while other complex elements have relatively simple data structures (e.g. AirDetail – consist of 3 simple elements and single level of nesting). Furthermore, additional coupling can take place among complex elements embedded within lower levels of the message structure. A more detail analysis will need to take into account simple elements by flattening XML structures and substituting all complex elements by corresponding simple data elements to give a more precise estimate of data coupling.

Acknowledgments

This research has been supported by GACR (Grant Agency, Czech Republic) grant No. P403/10/0092, University of Economics, Prague IGA (Internal Grant Agency) grant No. IG406040, and the Research Centre for Human Centered Technology Design at the University of Technology, Sydney, Australia.

References

1. Erl, T.: Service-oriented architecture: concepts, technology, and design. Prentice Hall PTR, Upper Saddle River (2005) ISBN: 0-13-185858-0
2. Hurwitz, J., Bloor, R., Baroudi, C.: Thinking from Reuse - SOA for Renewable Business (2006), http://www.hurwitz.com/PDFs/IBMThinkingfromReuse.pdf (cited December 13, 2007)
3. Sillitti, A., Vernazza, T., Succi, G.: Service Oriented Programming: A New Paradigm of Software Reuse. In: Gacek, C. (ed.) ICSR 2002. LNCS, vol. 2319, p. 269. Springer, Heidelberg (2002)
4. Blake, B.A., Jalics, P.: An Assessment of Object-oriented Methods and C++. Journal of Object-Oriented Programming 9(1), 42–48 (1996)
5. Fingar, P.: Component based frameworks for E-commerce. Communication of the ACM, 61–85 (2000)
6. Vitharana, P.: Risks and challenges of component based software development. Communications of the ACM 46(8), 67–72 (2003)
7. Natis, Y.V.: Service-Oriented Architecture Scenario (2005), http://www.gartner.com/resources/114300/114358/114358.pdf (cited May 3, 2008)
8. Meyer, B.: Object-oriented software construction, 2nd edn. Prentice Hall, Upper Saddle River (1997) ISBN: 0136291554
9. Fielding, R.T.: Architectural Styles and the Design of Network-based Software Architectures (2000), http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm
10. OTA. OTA Specifications (2010), http://www.opentravel.org/Specifications/Default.aspx (cited May 6, 2010)
11. Glushko, R., McGrath, T.: Document engineering: analyzing and designing documents for business informatics and Web services. MIT Press Books, Cambridge (2008)

12. Glushko, R., McGrath, T.: Patterns and reuse in document engineering. In: XML 2002 Proceedings (2002)
13. Glushko, R.J., McGrath, T.: Document Engineering for e-Business. In: Proceedings of the 2002 ACM symposium on Document Engineering (DocEng 2002), McLean, Virginia, USA. ACM Press, New York (2002)
14. ebXML. ebXML - Enabling A Global Electronic Market (2007), http://www.ebxml.org/ (cited December 9, 2007)
15. Feuerlicht, G., Lozina, J.: Understanding Service Reusability. In: 15th International Conference Systems Integration 2007. VSE Prague, Prague (2007) ISBN: 978-80-245-1196-2
16. Papazoglou, M.P., Yang, J.: Design Methodology for Web Services and Business Processes. In: Buchmann, A., Casati, F., Fiege, L., Hsu, M.-C., Shan, M.-C. (eds.) TES 2002. LNCS, vol. 2444, p. 54. Springer, Heidelberg (2002)
17. Papazoglou, M.P., Heuvel, W.V.D.: Service-oriented design and development methodology. International Journal of Web Engineering and Technology 2(4), 412–442 (2006)
18. Vinoski, S.: Old measures for new services. IEEE Internet Computing 9(6), 72–74 (2005)
19. Pautasso, C., Zimmermann, O., Leymann, F.: Restful web services vs. big'web services: making the right architectural decision. In: 17th International Conference on World Wide Web. ACM, Beijing (2008) ISBN: 978-1-60558-085-2
20. Pautasso, C., Wilde, E.: Why is the web loosely coupled?: a multi-faceted metric for service design. In: 18th International Conference on World Wide Web. ACM, Madrid (2009) ISBN: 978-1-60558-487-4
21. Page-Jones, M.: The Practical Guide to Structured Systems Design, 2nd edn. Prentice Hall, New Jersey (1988)
22. Feuerlicht, G.: Design of Service Interfaces for e-Business Applications using Data Normalization Techniques. Journal of Information Systems and e-Business Management, 1–14 (2005)
23. Stevens, W.P., Myers, G.J., Constantine, L.L.: Structured Design. IBM Systems Journal 38(2&3) (1999)
24. Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W.: Object-oriented modeling and design. Prentice Hall, New Jersey (1991)
25. Schmelzer: Solving the service granularity challenge (2007), http://searchsoa.techtarget.com/tip/0,289483,sid26_gci1172330,00.html (cited December 13, 2007)
26. Perepletchikov, M., Ryan, C., Frampton, K.: Comparing the Impact of Service-Oriented and Object-Oriented Paradigms on the Structural Properties of Software. In: Chung, S., Herrero, P. (eds.) OTM-WS 2005. LNCS, vol. 3762, pp. 431–441. Springer, Heidelberg (2005)
27. Feuerlicht, G.: System Development Life-Cycle Support for Service-Oriented Applications. In: 5th International Conference on Software Methodologies, Tools and Techniques, SoMet 2006. IOS Press, Quebec (2006) ISBN: 1-58603-673-4
28. Sindhgatta, R., Sengupta, B., Ponnalagu, K.: Measuring the Quality of Service Oriented Design. In: Baresi, L., Chi, C.-H., Suzuki, J. (eds.) ICSOC-ServiceWave 2009. LNCS, vol. 5900, pp. 485–499. Springer, Heidelberg (2009)
29. Chidamber, S., Kemerer, C.: A metrics suite for object oriented design. IEEE Transactions on Software Engineering 20(6), 476–493 (2002)
30. Perepletchikov, M., Ryan, C., Frampton, K.: Cohesion metrics for predicting maintainability of service-oriented software. In: QSIC, pp. 328–335 (2007)

31. Sindhgatta, R., Sengupta, B., Ponnalagu, K.: Measuring the Quality of Service Oriented Design. In: Baresi, L., Chi, C.-H., Suzuki, J. (eds.) ICSOC-ServiceWave 2009. LNCS, vol. 5900, pp. 485–499. Springer, Heidelberg (2009)
32. Sabre: Sabre Holdings: World Leader in Travel Distribution, Merchandising, and Retailing Airline Products (2007), http://www.sabre-holdings.com/ (cited 10 December 2007)
33. Alliance, O.T. OpenTravel™ Alliance XML Schema Design Best Practices (2010), http://www.opentravel.org/Resources/Uploads/PDF/OTA_SchemaDesignBestPracticesV3.06.pdf (cited September 1, 2010)
34. Feuerlicht, G., Meesathit, S.: Design framework for interoperable service interfaces. In: 2nd International Conference on Service Oriented Computing. ACM Press, New York (2004) ISBN: 1-58113-871-7

Wisdom-Aware Computing: On the Interactive Recommendation of Composition Knowledge

Soudip Roy Chowdhury, Carlos Rodríguez, Florian Daniel, and Fabio Casati

University of Trento, Via Sommarive 14, 38123 Povo (TN), Italy
{rchowdhury,crodriguez,daniel,casati}@disi.unitn.it

Abstract. We propose to enable and facilitate the development of service-based development by exploiting *community composition knowledge*, i.e., knowledge that can be harvested from existing, successful mashups or service compositions defined by other and possibly more skilled developers (the *community* or *crowd*) in a same domain. Such knowledge can be used to assist less skilled developers in defining a composition they need, allowing them to go beyond their individual capabilities. The assistance comes in the form of *interactive advice*, as we aim at supporting developers while they are defining their composition logic, and it adjusts to the skill level of the developer. In this paper we specifically focus on the case of *process-oriented, mashup-like applications*, yet the proposed concepts and approach can be generalized and also applied to generic algorithms and procedures.

1 Introduction

Although each of us develops and executes various procedures in our daily life (examples range from cooking recipes to low-level programming code), today very little is done to support others, possibly *less skilled developers* (or, in the extreme case, even end users) in developing their own. Basically, there are two main approaches to **enable less skilled people** to "develop": either development is eased by *simplifying* it (e.g., by limiting the expressive power of a development language) or it is facilitated by *reusing knowledge* (e.g., by copying and pasting from existing algorithms).

Among the **simplification** approaches, the workflow and BPM community was one of the first to claim that the abstraction of business processes into tasks and control flows would allow also the less skilled users to define own processes, however with little success. Then, with the advent of web services and the service-oriented architecture (SOA), the web service community substituted tasks with services, yet it also didn't succeed in enabling less skilled developers to compose services. Recently, web mashups added user interfaces to the composition problem and again claimed to target also end users, but mashup development is still a challenge for skilled developers. While these attempts were aimed at simplifying technologies, the human computer interaction community has researched on end user development approaching the problem from the user interface perspective. The result is simple applications that are specific to a very limited domain, e.g., an interactive game for children, with typically little support for more complex applications.

E.M. Maximilien et al. (Eds): ICSOC 2010 Workshops, LNCS 6568, pp. 144–155, 2011.

As for what regards **capturing and reusing knowledge**, in IT reuse typically comes in the form of program libraries, services, or program templates (such as generics in Java or process templates in workflows). In essence, what is done today is either providing building blocks that can be composed to achieve a goal, or providing the entire composition (the algorithm – possibly made generic if templates are used), which may or may not suit a developer's needs. In the nineties and early 2000s, AI planning [1] and automated, goal-oriented compositions (e.g., as in [2]) became popular in research. A typical goal there is to derive a service composition from a given goal and a set of components and composition rules. Despite the large body of interesting research, this thread failed to produce widely applicable results, likely because the goal is very ambitious and because assumptions on the semantic richness and consistency of component descriptions are rarely met in practice. Other attempts to extract knowledge are, for example, oriented at identifying social networks of people [3] or at providing rankings and recommendations of objects, from web pages (Google's Pagerank) to goods (Amazon's recommendations). An alternative approach is followed by expert recommender systems [4], which, instead of identifying knowledge, aim at identifying knowledge holders (the experts), based on their code production and social involvement.

In this paper, we describe **WIRE**, a *WIsdom-awaRE development environment* we are currently developing in order to enable less skilled developers to perform also complex development tasks. We particularly target process-oriented, mashup-like applications, whose development and execution can be provided as a service via the Web and whose internals are characterized by relatively simple composition logic and relatively complex tasks or components. This class of programs seems to provide both the benefit of (relative) simplicity and a sufficient information base (thanks to the reuse of components) to learn and reuse programming/service composition knowledge. The idea is to *learn from existing compositions* (or, in general, *computations*) and to provide the learned knowledge in form of *interactive advice* to developers while they are composing their own application in a visual editor. The aim is both to allow developers to go beyond their own development capabilities and to speed up the overall development process, joining the benefits of both simplification *and* reuse.

Next, we discuss a state of the art composition scenario and we show that it is everything but trivial. In Section 3, we discuss the state of the art in assisted composition. In Section 4 and 5, we investigate the idea of composition advices and provide our first implementation ideas, respectively. Then we conclude the paper and outline our future work.

2 Example Scenario and Research Challenges

In order to better understand the problem we want to address, let's have a look at how a mashup is, for instance, composed in Yahoo! Pipes (*http://pipes.yahoo.com/pipes/*), one of the most well-known mashup platforms as of today. Let's assume we want to develop a simple pipe that sources a set of news from *Google News*, filters them according to a predefined filter condition (in our case, we want to search for news on products and services by a given vendor), and locates them on a *Yahoo! Map*.

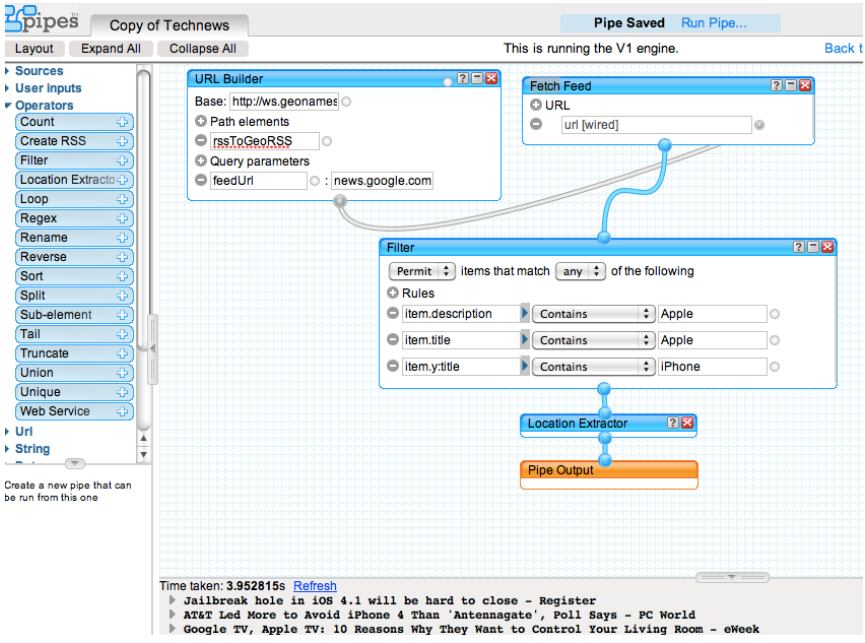


Fig. 1. Implementation of the example scenario in Yahoo! Pipes

The pipe that implements the required feature is illustrated in Figure 1. It is composed of five components: The *URL Builder* is needed to set up the remote *Geo Names* service, which takes a news RSS feed as an input, analyzes its content, and inserts geo-coordinates, i.e., longitude and latitude, into each news item (where possible). Doing so requires setting some parameters: *Base=http://ws.geonames.org*, *Path elements=rssToGeoRSS*, and *Query parameters=FeedUrl:news.google.com/news?topic=t&output=rss&ned=us*. The so created URL is fed into the *Fetch Feed* component, which loads the geo-enriched news feed. In order to filter out the news items we are really interested in, we need to use the Filter component, which requires the setting of proper filter conditions via the *Rules* input field. Feeding the filtered feed into the *Location Extractor* component causes Pipes to plot the news items on a *Yahoo! Map*. Finally, the *Pipe Output* component specifies the end of the pipe.

If we analyze the development steps above, we can easily understand that developing even such a simple composition is out of the reach of people without programming knowledge. Understanding which components are needed and how they are used is neither trivial nor intuitive. The URL Builder, for example, requires the setting of some complex parameters. Then, components need to be suitably connected, in order to support the data flow from one component to another, and output parameters must be mapped to input parameters. But more importantly, plotting news onto a map requires knowing that this can be done by first enriching a feed with geo-coordinates, then fetching the actual feed, and only then the map is ready to plot the items.

Enabling non-expert developers to compose a pipe like the above requires telling (or teaching) them the necessary knowledge. In **WIRE**, we aim to do so by providing

non-expert developers with interactive development advices for composition, inside an assisted development environment. We want to obtain the knowledge to provide advices by extracting, abstracting, and reusing compositional knowledge from existing compositions (in the scenario above, pipes) that contain community knowledge, best practices, and proven patterns. That is, in *WIRE* we aim at bringing the *wisdom of the crowd* (possibly even a small crowd if we are reusing knowledge within a company) in defining compositions when they are both defined by an individual (where the crowd supports an individual) or by a community (where the crowd supports social computing, i.e., itself in defining its own algorithms). The final goal is to move towards a new frontier of knowledge reuse, i.e., *reuse of computational knowledge*.

Doing so requires approaching a set of **challenges** that are non-trivial:

1. First of all, *identifying the types of advices that can be given and the right times when they can be given*: depending on the complexity and expressive power of the composition language, there can be a huge variety of possible advices. Understanding which of them are useful is crucial to limit complexity.
2. *Discovering computational knowledge*: how do we harvest development knowledge from the crowd, that is, from a set of existing compositions? Knowledge may come in a variety of different forms: component or service compatibilities, data mappings, co-occurrence of components, design patterns, evolution operations, and so on.
3. *Representing and storing knowledge*: once identified, how do we represent and store knowledge in a way that allows easy querying and retrieval for reuse?
4. *Searching and retrieving knowledge*: given a partial program specification under development, how do we enable the querying of the knowledge space and the identification of the most suitable and useful advice to provide to the developer, in order to really assist him?
5. *Reusing knowledge*: given an advice for development, how do we (re)use the identified knowledge in the program under development? We need to be able to "weave" it into the partial specification in a way that is correct and executable, so as to provide concrete benefits to the developer.

In this paper, we specifically focus on the *first challenge* and we provide our first ideas on the second challenge and on the assisted development environment.

3 State of the Art

In **literature**, there are approaches that aim at similar goals as WIRE, yet they mainly focus on the *retrieval* and *reuse* of composition knowledge. In [6], for instance, mashlets (the elements to be composed) are represented via their inputs and outputs, and glue patterns are represented as graphs of connections among them; reuse comes in the form of auto-completion of missing components and connections, selected by the user from a ranked list of top-k recommendations obtained starting from the mashlets used in the mashup. In [8], light-weight semantic annotations for services, feeds, and data flows are used to support a text-based search for data mashups, which are actually generated in an automated, goal-oriented fashion using AI planning (the search tags are the goals); generated data processing pipes can be used as is or further edited.

The approach in [9] semantically annotates portlets, web apps, widgets, or Java beans and supports the search for functionally equivalent or matching components; reuse is supported by a semantics-aided, automated connection of components. Also the approach in [10] is based on a simple, semantic description of information sources (name, formal inputs [allowed ones], actual inputs [outputs consumed from other sources], outputs) and mashups (compositions of information sources), which can be queried with a partial mashup specification in order identify goals based on their likelihood to appear in the final mashup; goals are fed to a semantic matcher and an AI planner, which complete the partial mashup. This last approach is the only one that also automatically *discovers* some form of knowledge in terms of popularity of outputs in existing mashup specifications (used to compute the likelihoods of goals).

In the context of business process modeling, there are also some works with similar goals as ours. For instance, in [7], the authors more specifically focus on business processes represented as Petri nets with textual descriptions, which are processed (also leveraging WordNet) to derive a set of descriptive tags that can be used for search of processes or parts thereof; reuse is supported via copy and paste of results into the modeling canvas. The work presented in [13] proposes an approach for supporting process modeling through object-sensitive action patterns, where these patterns are derived from a repository of process models using techniques from association rule learning, taking into consideration not only actions (tasks), but also the business objects to which these actions are related. Finally, [14] presents a model for the reuse data mining processes by extending the CRISP-DM process [15]. The proposed model aims at including data mining process patterns into CRISP-DM and to guide the specialization and application of such patterns to concrete processes, rather than actually exploiting the community knowledge.

In general, the **discovery of community composition knowledge** is not approached by the works above (or they do it in a limited way, e.g., by deriving only behavioral patterns from process definitions). Typically, they start from an annotated representation of mashups and components and query them for functional compatibilities and data mappings, improving the quality of search results via semantics, which are explicit and predefined. WIRE, instead, specifically focuses on the elicitation and collection of *crowd wisdom*, i.e., composition knowledge that derives from the ways other people have solved similar composition problems in the past and that has a significant support in terms of number of times it has been adopted. This means that in order to create knowledge for WIRE, we do not need any expert developer or domain specialist that writes and maintains explicit composition rules or logics; knowledge is instead harvested from how people compose their very own applications, without requiring them to provide additional meta-data or descriptions (which typically doesn't work in practice).

4 Wisdom-Aware Development: Concepts and Principles

Identifying which advices can be provided and which advices do indeed have the potential to help less skilled developers to perform complex development tasks requires, first of all, understanding the *expressive power* of the composition language at hand. We approach this task next. Then we focus on the advices.

4.1 Expressiveness of the Composition Language

Let us consider again Yahoo! Pipes. The platform has a very advanced and pleasant user interface for drag-and-drop development of data mashups and supports the composition of also relatively complex processing logics. Yet, the strong point of Pipes is its *data flow based composition paradigm*, which is very effective and requires only a limited set of modeling constructs. As already explained in the introduction, constraining the expressive power of composition languages is one of the techniques to simplify development, and Pipes shares this characteristic with most of today's mashup platforms.

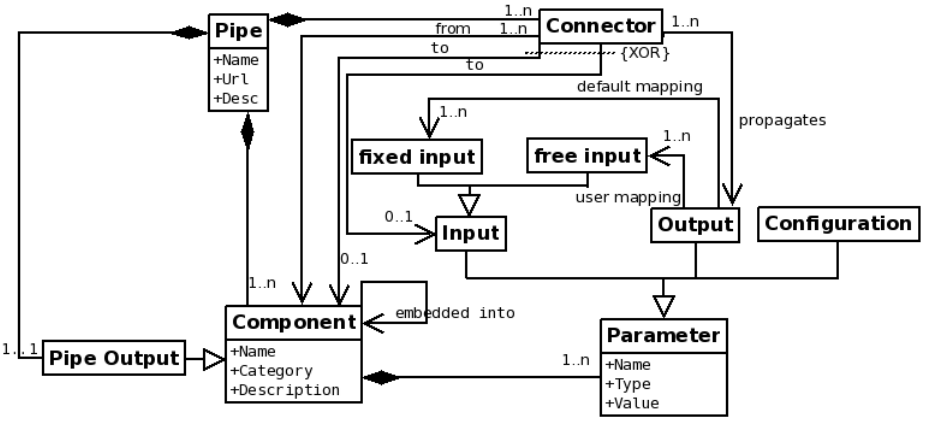


Fig. 2. A meta-model for Yahoo! Pipes' composition language

In order to better understand the expressiveness of Yahoo! Pipes, in Figure 2 we derived a **meta-model** for its composition language. A *Pipe* is composed of components and connectors. *Components* have a name and a description and may be grouped into categories (e.g., source components, user input components, etc.). Each pipe contains always one *Pipe Output* component, i.e., a special component that denotes the end of data flow logic or the end of the application. A component may be *embedded* into another component; for example components (except user inputs and operators) can be embedded inside a *Loop Operator* component. Components may also have a set of parameters. A *Parameter* has a name, a type, and may have a value assigned to it. There are basically three types of parameters: *input* parameters (accept data flows attributes), *output* parameters (produce data flow attributes), and *configuration* parameters (are manually set by the developer). For instance, in our example in Section 2, the *URL* parameter of the *Fetch Feed* component is an input parameter; the *longitude* and *latitude* attributes of the RSS feed fetched by the *Fetch Feed* component are output parameters; and the *Base* parameter of the *URL Builder* component is an example of configuration parameter.

Data flows in Pipes are modeled via dedicated connectors. A *Connector* propagates output parameters of one component (indicated in Figure 2 by the *from* relationship) to either another component or to an individual input field of another component. If a connector is connected to a whole component (e.g., in the case of the connector from

the *Fetch Feed* component to the *Filter* component in Figure 1), all attributes of the RSS item flowing through the connector can be used to set the values of the target component's input parameters. If a connector is connected only to a single input parameter, the data flow's attributes are available only to set the value of the target input parameter. Input parameters are of two types: either they are *fixed inputs*, for which there are predefined *default mappings*, or they are *free inputs*, for which the user can provide a value or choose which flow attribute to use. That is, for free inputs it is possible to specify a simple attribute-parameter data mapping logic.

Figure 2 shows that Yahoo! Pipes' meta-model is indeed very **simple**: only 10 concepts suffice to model its composition features. Of course, the focus of Pipes is on data mashups, and there is no need for complex web services or user interfaces, two features that are instead present in our own mashup platform, i.e., mashArt [5]. Yet, despite these two additions, mashArt's meta-model only requires 13 concepts. If instead we look at the BPMN modeling notation for business processes [11], we already need more than 20 concepts to characterize its expressive power, and the meta-model of BPEL [12] has almost 60 concepts! Of course, the higher the complexity of the language, the more difficult it is to identify and reuse composition knowledge.

4.2 Advising Composition Knowledge

Given the meta-model of the composition language for which we want to provide composition advices, it is possible to identify which concrete **compositional knowledge** can be extracted from existing compositions (e.g., pipes). The gray boxes in the conceptual model in Figure 3 illustrate the result of our analysis. The figure identifies the key entities and relationships needed to provide composition advices.

An *Advice* provides composition knowledge in form of composition patterns. An advice can be to *complete* a given pattern (given it's partial implementation in the modeling canvas) or to *substitute* a pattern with a similar one, or the advice can *highlight* compatible elements in the modeling canvas or *filter and rank* advices.

Patterns represent the actual recommendation that we deliver to the user. They can be of five types (all these patterns can be identified in the model in Figure 3):

- *Parameter Value Patterns*: Possible values for a given parameter. For instance, in the *URL Builder* component the *Base* parameter value in a pattern can be set to "http://ws.geonames.org", while the *Path elements* parameter value can be "rssToGeoRSS", and feedUrl can be "news.google.com/news?topic=t&output=rss&ned=us", as shown in our example scenario. Alternatively, we can have the *URL Builder* component with the *Base* parameter set to "news.google.com/news" and the *Query parameters* set with different values.
- *Component Association Patterns*: Co-occurrence patterns for pairs of components. For instance, in our scenario, whenever a user drags and drops the *URL Builder* on the design canvas, a possible advice derived from a component association pattern can be to include in the composition the *Fetch Feed* component and connect it to the *URL Builder*.
- *Connector Patterns*: Component-component or component-input parameter patterns. This pattern captures the dataflow logic, i.e., how components are connected via connector elements. For example, *URL Builder – connector-Fetch Feed* is a connector pattern in our example scenario.

- *Data Mapping Patterns*: Associations of outputs to inputs. In Figure 1, for instance, we map the *description*, *title*, and *y:title* attributes of the fetched feed to the first input field of the first, second, and third rule, respectively, telling the *Filter* component how we map the individual attributes in input to the individual, free input parameters of the component.
- *Complex Patterns:* Partial compositions consisting of multiple components, connectors, and parameter settings. In our example scenario, different combinations of components and connectors, having their parameter values set and with proper data mappings, as a part and as a whole represent complex patterns. For example, the configuration *URL Builder – Fetch Feed – Filter – Location Extractor*, along with their settings, represents a complex pattern.

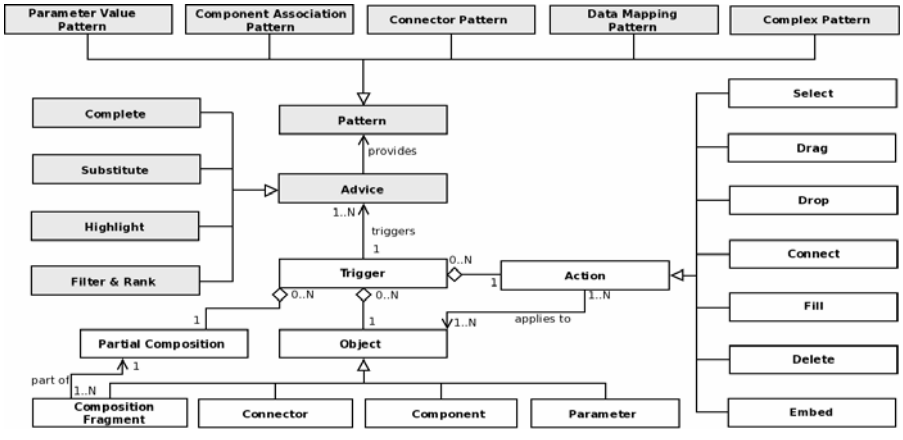


Fig. 3. Conceptual model of WIRE's advice approach. Gray entities model the ingredients for advices; white boxes model the advice triggering logic inside the design environment.

An *Advice* provides composition knowledge in form of composition patterns. An advice can be to *complete* a given pattern (given it's partial implementation in the modeling canvas) or to *substitute* a pattern with a similar one, or the advice can *highlight* compatible elements in the modeling canvas or *filter and rank* advices.

Patterns represent the actual recommendation that we would like to deliver to the user. They can be of five different types: *Complex Patterns* (partial compositions possibly consisting of multiple components, connectors, and parameter settings), *Parameter Value Patterns* (possible values for a given parameter), *Component Association Patterns* (co-occurrence patterns for pairs of components), *Connector Patterns* (component-component or component-input parameter patterns), and *Data Mapping Patterns* (associations of outputs to inputs).

Now, let us discuss the "white part" of the model. This part represents the entities that jointly define the conditions under which advices can be triggered. A *Trigger* for an advice is defined by an object, an action of the user in the modeling canvas, and the state of the current composition, i.e., the partial composition in the modeling canvas. This association can be thought of as a triplet that defines the triggering condition. The *Objects* a user may operate are *Composition Fragments* (e.g., a selection of

a subset of the pipe in the canvas), individual *Components*, *Connectors*, or *Parameters* (by interacting with the respective graphical input fields). The *Action* represents the action that the user may perform on an object during composition. We identify seven actions: *Select* (e.g., a composition fragment or a connector), *Drag* (e.g., a component or a connector endpoint), *Drop*, *Connect*, *Fill* (a parameter value), *Delete*, and *Embed* (one component into another). Finally, the *Partial Composition* represents the status of the current overall composition.

While the object therefore identifies *which* advice may be of interest to the user, the action decides *when* the advice can be given, and the state *filters* out advices that are not compatible with the current partial composition (e.g., if the *Location Extractor* component has already been used, recommending its use becomes useless).

Regarding the model in Figure 3, not all associations may be needed in practice. For instance, not all components are compatible with the *embed* action. Yet, the model identifies precisely which advices can be given and when.

5 The WIRE Platform

Figure 4 illustrates the high-level architecture of the assisted development environment with which we aim at supporting wisdom-aware development according to the model described in the previous section: developers can design their applications in a *wisdom-aware development environment*, which is composed of an *interactive recommender* (for development advice) and an *offline recommender* as well as the *wisdom-aware editor* implementing the interactive development paradigm. Compositions or mashups are stored in a *compositions repository* and can be executed in a dedicated *runtime environment*, which generates *execution data*. Compositions and execution data are the input for *the knowledge/advice extractor*, which finds the repeated and useful patterns in them and stores them as development and evolution advice in the *advice repository*. Then, the *recommenders* provide them as interactive advices through its query interface upon the current context and triggers of the user's development environment. Here, we specifically focused on development advices related to composition; we will approach evolution advices in our future work (evolution advices will, for instance, take into account performance criteria or evolutions applied by developers over time on their own mashups).

We realize that each domain will have suitable languages and execution engines, such as a mashup engine or a scientific workflow engine. Our goal is not to compete with these, but to define mechanism to "WIRE" these languages and tools with the ability to extract knowledge and provide advice. For this reason, in this paper we started with studying the case of Yahoo! Pipes, which is well known and allows us to easily explain our ideas. We however intend to apply the wisdom-aware development paradigm to our own mashup editor, mashArt [5], which features a *universal composition* paradigm user interface components, application logic, and data web services, a development paradigm that is similar in complexity to that of Pipes.

As for the reuse of knowledge, the **WIRE approach** is not based on semantic annotations, matching, or AI planning techniques, nor do we aim at automated or goal-driven composition or at identifying semantic similarity among services. We also do not aim at having developers tag components or add metadata to let others better

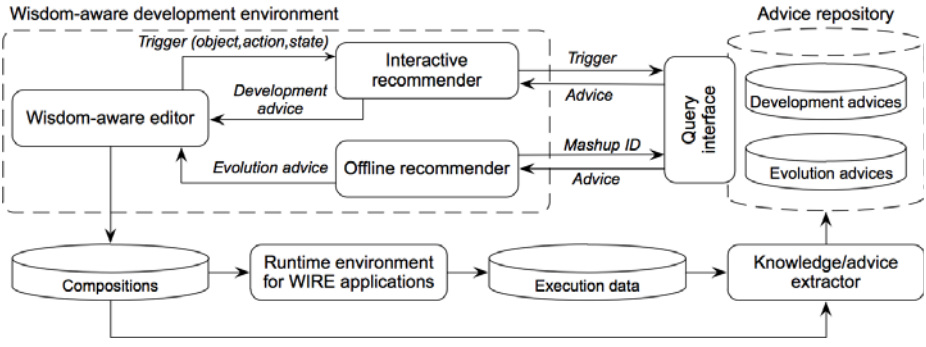


Fig. 4. High-level architecture of the envisioned system for wisdom-aware development

reuse services, processes, or fragments. In other words, we aim at collecting knowledge *implicitly*, as we believe that otherwise we would face an easier wisdom extraction problem but end up with a solution that in practice does not work because people do not bother to add the necessary metadata. WIRE will rather leverage on statistical data analysis techniques and data mining as means to extract knowledge from the available information space. To do so, we propose the following core steps:

1. *Cleaning, integration, and transformation:* We take as input previous compositions and execution data and prepare them for the analysis.
2. *Statistical data analysis and data mining:* On the resulting data, we apply statistical data analysis and data mining techniques, which may include mining of frequent patterns, association rules, correlations, classification and cluster analysis. The results of this step are used to create the composition patterns.
3. *Evaluation and ranking of advices (knowledge):* Once we have discovered the potential advices, we evaluate and rank them using standard interestingness measures (e.g., support and confidence) and ranking algorithms.
4. *Presentation of advices:* The advices are presented to the user through intuitive visual metaphors that are suitable to the context and purpose of the advice.
5. *Gathering of user feedback:* The popularity of advices is gathered and measured in order to better rank them.

Among the techniques we are applying for the discovery tasks, we are specifically leveraging on data mining approaches, such as *frequent itemset mining*, *association rules learning*, *sequential pattern mining*, *graph mining,* and *link mining*. Each of these techniques can be used to discover a different type of advice:

- *Frequent itemset mining:* The objective of this technique is to find the co-occurrence of items in a dataset of transactions. The co-occurrence is considered "frequent" whenever its support equals or exceeds a given threshold. This technique can be used as a support for discovering any of the advices introduced before. For instance, in the case of discovering *Component Association Patterns* we can this technique.
- *Association rules:* This technique aims at finding rules of the form $A \rightarrow B$, where A and B are disjoint sets of items. This technique can be applied to help in the discovery of any of the proposed advices. For instance, in the case of the

Parameter Value Pattern, given the value of two parameters of a component, we can find an association rule that suggests us the value for a third parameter.

- *Sequential pattern mining:* Given a dataset of sequences, the objective of sequential pattern mining is to find all sequences that have a support equal or greater than a given threshold. This technique can be applied to discover *Complex Patterns*, *Component Association Patterns,* and *Connector Patterns*. For instance, in the case of the *Connector Pattern*, we can use this technique to extract patterns that can be then used for suggesting connectors among components placed on the design canvas.
- *Graph mining:* given a set of graphs, the goal of graph mining is to find all subgraphs such that their support is equal or greater than a given threshold. For our purpose, we can use graph mining for discovering *Complex Patterns* and *Connector Patterns*. For instance, for *Complex Patterns* we can suggest a list of existing ready compositions based on the partial composition the user has in the canvas, whenever this partial composition is deemed as frequent.
- *Link mining:* rather than a technique, link mining refers to a set of techniques for mining data sets where objects are linked with rich structures. Link mining can be applied to support the discovery of any of the proposed advices. For example, in the case of *Data Mapping Patterns*, we can discover patterns for mapping the parameters of two components, based on the types these parameters.

Once community composition knowledge has been identified, we store the extracted knowledge in the advice repository in the form of directed graphs. In our advice repository, elements in the patterns, e.g., a component or a connector, are represented as nodes of the graph, and relationships among them, e.g., a component "has" a parameter, are represented as edges between those nodes. We also store a set of rules in our advice repository, which represent the trigger conditions under which a specific knowledge can be provided as an advice. Based upon this information, through our query interface we can match knowledge with the current composition context and retrieves relevant advices from the advice repository. Retrieved advices are filtered, ranked, and delivered based on user profile data (e.g., the programming expertise of the user or his/her preferences over advice types).

6 Conclusion

In this paper we propose the idea of *wisdom-aware computing*, a computing paradigm that aims at *reusing community composition knowledge* (the wisdom) to provide interactive development advice to less skilled developers. If successful, WIRE can e*xtend the "developer base"* in each domain where reuse of algorithmic knowledge is possible and it can *facilitate progressive learning and knowledge transfer*.

Unlike other approaches in literature, which typically focus on *structural and semantic similarities*, we specifically focus on the elicitation of composition knowledge that derives from the expertise of people and that is expressed in the compositions they develop. If, for instance, two components have been used together successfully multiple times, very likely their joint use is both syntactically and semantically meaningful. There is no need to further model complex ontologies or composition rules.

In order to provide identified patterns with the necessary semantics, we advocate the application of the WIRE paradigm to composition environments that focus on

specific domains. Inside a given domain, component names are self-explaining and patterns can easily be understood. In the Omelette (*http://www.ict-omelette.eu/*) and the LiquidPub (*http://liquidpub.org/)* projects, we are, for instance, working on two domain-specific mashup platforms for telco and research evaluation, respectively.

For illustration purposes, in this paper we used Yahoo! Pipes as reference mashup platform, as Pipes is very similar in complexity to our own *mashArt* platform [5] but better known. In order to have access to the compositions that actually hold the knowledge we want to harvest, we will of course apply WIRE to mashArt.

Acknowledgements. This work was supported by funds from the European Commission (project OMELETTE, contract no. 257635).

References

1. Geffner, H.: Perspectives on artificial intelligence planning. In: AAAI 2002, pp. 1013–1023 (2002)
2. Roman, D., de Bruijn, J., Mocan, A., Lausen, H., Domingue, J., Bussler, C., Fensel, D.: WWW: WSMO, WSML, and WSMX in a nutshell. In: Mizoguchi, R., Shi, Z.-Z., Giunchiglia, F. (eds.) ASWC 2006. LNCS, vol. 4185, pp. 516–522. Springer, Heidelberg (2006)
3. Koschmider, A., Song, M., Reijers, H.A.: Social Software for Modeling Business Processes. In: Ardagna, D., Mecella, M., Yang, J. (eds.) BPM 2008 Workshops. LNBIP, vol. 17, pp. 666–677. Springer, Heidelberg (2009)
4. Reichling, T., Veith, M., Wulf, V.: Expert Recommender: Designing for a Network Organization. Computer Supported Cooperative Work 16(4-5), 431–465 (2007)
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. Greenshpan, O., Milo, T., Polyzotis, N.: Autocompletion for mashups. In: VLDB 2009, pp. 538–549 (2009)
7. Hornung, T., Koschmider, A., Lausen, G.: Recommendation based process modeling support: Method and user experience. In: Li, Q., Spaccapietra, S., Yu, E., Olivé, A. (eds.) ER 2008. LNCS, vol. 5231, pp. 265–278. Springer, Heidelberg (2008)
8. Riabov, A.V., Bouillet, E., Feblowitz, M.D., Liu, Z., Ranganathan, A.: Wishful Search: Interactive Composition of Data Mashups. In: WWW 2008, pp. 775–784 (2008)
9. Ngu, A.H.H., Carlson, M.P., Sheng, Q.Z.: Semantic-Based Mashup of Composite Applications. IEEE Transactions on Services Computing 3(1) (January-March 2010)
10. Elmeleegy, H., Ivan, A., Akkiraju, R., Goodwin, R.: MashupAdvisor: A Recommendation Tool for Mashup Development. In: ICWS 2008, pp. 337–344 (2008)
11. OMG. Business Process Model and Notation (BPMN) - Version 1.2 (January 2009), `http://www.omg.org/spec/BPMN/1.2`
12. OASIS. Web Services Business Process Execution Language Version 2.0 (April 2007), `http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html`
13. Smirnov, S., Weidlich, M., Mendling, J., Weske, M.: Object-Sensitive Action Patterns in Process Model Repositories. In: BPM 2010 Workshops, NJ, USA (September 2010)
14. Wegener, D., Rueping, S.: On Reusing Data Mining in Business Processes – A Pattern-based Approach. In: BPM 2010 Workshops, NJ, USA (September 2010)
15. Shearer, C.: The CRISP-DM model: the new blueprint for data mining. Journal of Data Warehousing 5(4), 13–22 (2000)

Introduction to the First International Workshop on Services, Energy, and Ecosystem (SEE 2010)

Barbara Pernici[1], Schahram Dustdar[2], G.R. Gangadharan[1], Patricia Lago[3], and San Murugesan[4]

[1] Politecnico di Milano, Italy
barbara.pernici@polimi.it, geeyaar@gmail.com
[2] Technical University of Vienna, Austria
[3] VU University Amsterdam, Netherlands
patricia@cs.vu.nl
[4] University of Western Sydney, Australia
san1@internode.net

Abstract. The first international workshop on Services, Energy, and Ecosystem (SEE 2010) focused on creating sustainable (green) energy-efficient services and fostering the growth towards a new eco-friendly world of services. Technical papers of high quality have been submitted, of which six were accepted (with a 60% of acceptance rate). Moreover, a keynote given by Professor Barbara Pernici presented an overview on research projects funded by the European Commission toward energy efficiency in ICT.

Keywords: green IT, energy efficiency, services.

1 Aims and Scope

Although significant progress has been made in the recent years in making computing greener, i.e. energy-efficient and environmentally sustainable, the net impact of improvements in energy efficiency is generally more than offset by increasing demand for computing power and capacity, driven by new digitized business processes and services. The total energy consumption has been increasing due to proliferation of computers, data centers and various types of mobile computing and communication devices, as well as by the emergence of, and demand for, new applications. To address this issue holistically, there is now call for making applications and services themselves energy-aware and -efficient. New innovative ICT-based tools for monitoring and managing energy consumption in several application have emerged, and novel approaches for creating energy-aware systems and applications are being developed to create a better and sustainable ecosystem.

The SEE workshop solicited research submissions and real-world experiences on all topics related to Services, Energy, and Ecosystems, to bring together

E.M. Maximilien et al. (Eds): ICSOC 2010 Workshops, LNCS 6568, pp. 156–157, 2011.

researchers and practitioners from multidisciplinary fields working on energy-aware and energy-efficient systems and applications and to set an agenda for further work in this important new area. In particular the following topics have been presented and discussed:

1. Engineering Energy-Efficient Systems, Applications and Services;
2. Monitoring and Managing Energy-Aware Services;
3. Fostering Ecosystems in Organizations and Society.

2 Workshop Co-organizers

- Prof. Barbara Pernici (Politecnico di Milano, Italy)
- Prof. Schahram Dustdar (Technical University of Vienna, Austria)
- Dr. G.R. Gangadharan, Politecnico di Milano, Italy)
- Prof. Patricia Lago, VU University Amsterdam, Netherlands)
- Prof. San Murugesan (University of Western Sydney, Australia)

3 Programm Committee

- Rami Bahsoon University of Birmingham, UK
- Ivona Brandic, Technical Univ. of Vienna, Austria
- Jon Crowcroft, Cambridge University, UK
- Mariagrazia Fugini, Politecnico di Milano, Italy
- Aditya Ghose, University of Wollongong, Australia
- Ronen Kat, IBM Haifa, Israel
- Alexander Kipp, University of Stuttgart, Germany
- Dana Petcu, West University of Timisoara, Romania
- Pierluigi Plebani, Politecnico di Milano, Italy
- Ioan Salomie, Tech. Univ. of Cluj-Napoca, Romania
- Thomas Setzer, Tech. Univ. of Mnchen, Germany
- Bhuvan Unhelkar, MethodScience, Australia

4 Additional Reviewers

- Ivan Breskovic

A Dynamic Power Management Controller for Optimizing Servers' Energy Consumption in Service Centers

Tudor Cioara[1], Ioan Salomie[1], Ionut Anghel[1], Iulian Chira[1], Alexandru Cocian[1], Ealan Henis[2], and Ronen Kat[2]

[1] Technical University of Cluj-Napoca, Computer Science Department, Cluj-Napoca, Romania
{Tudor.Cioara,Ioan.Salomie,Ionut.Anghel}cs.utcluj.ro
[2] IBM Haifa Research Laboratory, Haifa, Israel
{Ealan,Ronenkat}@il.ibm.com

Abstract. This paper proposes the development of a management controller, which balances the service center servers' workload and hardware resources usage to locally optimize energy consumption. The controller exploits energy saving opportunities due to short-term fluctuations in the performance request levels of the server running tasks. The paper proposes Dynamic Power Management strategies for processor and hard disks which represent the main elements of the controller energy consumption optimization process. We propose techniques for identifying the over-provisioned resources and putting them into low-power states until there is a prediction for a workload requiring scaling-up the server's computing capacity. Virtualization techniques are used for a uniform and dependence free management of server tasks.

Keywords: Dynamic Power Management, Service Center Server, Energy Efficiency.

1 Introduction and Related Work

The past decade in the field of computing has been marked by the advent of web-based applications and online services, which translated into an exponential growth of the underlying service center infrastructure. A 2007 report [1] to Congress, by the U.S. Environmental Protection Agency, shows that in just five years, the electricity consumed by service centers and their additional infrastructure will be doubled and the trend is expected to accelerate driven by the shift towards cloud computing.

The GAMES (Green Active Management of Energy in IT Service Centers) [2] research project aims at developing a set of innovative methodologies, metrics, services and tools for the active management of energy efficiency of IT service centers. In the GAMES project we approach the service center energy efficiency problem by using two types of energy aware control loops: a global control loop associated with the entire service center and a set of local control loops associated with each service

E.M. Maximilien et al. (Eds): ICSOC 2010 Workshops, LNCS 6568, pp. 158–168, 2011.

center server. The global control loop uses service center energy/ performance data for taking run-time adaptation decisions to enforce the predefined Green and Key Performance Indicators (GPIs / KPIs). The local control loops balance the service center servers' workload and hardware resources usage to locally optimize the servers' energy consumption, without considering the entire service center state.

In this paper we propose the development of an autonomic management controller, which targets the objectives of the local control loop, for a virtualized server of a service center. The controller, also called the Local Loop Controller, manages virtualized tasks at the server level so that the performance levels required by the tasks running on the server are achieved and maintained while maximizing the server energy efficiency. The controller exploits energy saving opportunities presented by short-term fluctuations in the performance request levels of the server running tasks. We use virtualization as an abstraction level, because it allows us to manage the tasks in a uniform manner, without worrying about application dependencies and low-level details. The virtualized tasks are annotated with Quality-of-Service (QoS) requirements representing the tasks' performance request levels. Local Loop Controller energy consumption optimization process is based on Dynamic Power Management (DPM) actions aiming at putting the over-provisioned resources into low-power states. The resources remain in low-power states until there is a predictive workload that requires scaling-up the server's computing capacity.

The service center servers' energy efficiency is a complex issue and existing research covers a wide variety of techniques which have to be taken in a holistic approach. It involves concepts such as resource provisioning and allocation, virtualization and dynamic power management. Khargharia [3] proposes a theoretical methodology and an experimental framework for autonomic power and performance management of high-performance server platforms. The over-provisioned server resources are transitioned to low-power states until there is an increase in the workload that would require scaling-up the server capacity again. Similarly, in [4] the performance-power management problem is decomposed into smaller sub-problems implemented locally by individual controllers within each hardware component. A fuzzy resource allocation approach where a fuzzy logic controller is used to dynamically adjust the number of resources needed in serving requests is presented in [5]. The main problem with DPM techniques is that changing between power states has performance implications, and sometimes energy penalties [6]. Algorithms have been developed for DPM by considering the power/performance characteristics of main hardware devices, mainly the processor and the hard disk drive and, to some extent, the system memory (mostly hardware controllers) [7]. Bircher and John [8] perform an ample analysis of power management on recent multi-core processors using the functions provided by regular operating systems. For hard drive power management Chung [9] proposes an active-idle states transition method based on adaptive learning tree in which idle and active periods are stored as tree nodes. In [10], [11] and [14] power-aware storage cache management algorithms are presented. They also provide opportunities for the underlying disk power management schemes to save energy, such as off-line greedy algorithms and on-line cache write policies. The use of virtualization together with DPM techniques raises serious challenges due the lack of power metering for virtual machines. Unlike physical servers that provide in-hardware or outlet level power measurement functions, it is very difficult to estimate the power

required by a virtual machine when hosted on a certain hardware configuration [12]. The performance and health monitoring counters exposed by the virtual machine hypervisor, correlated with relevant power information about the hosting hardware and simple adaptive power models can offer relatively accurate information [13].

The rest of this paper is organized as follows: Section 2 presents the architecture of the Local Loop Controller, Section 3 describes the proposed DPM algorithms, Section 4 shows a the experimental results, while Section 5 concludes the paper.

2 Local Loop Controller Architecture

The Local Loop Controller acts at the server level with the goal of maximizing server energy efficiency while preserving the performance levels required by the running tasks. From a power management perspective, we emphasize that the most power-hungry components within a server are the CPU, the HDD (Hard Disk Drive) and, to some extent, the internal memory. The Local Loop Controller exploits the non-mutually exclusive energy / performance related behavior of these components and sets them to low-power states. The workload is composed of virtualized tasks annotated with Quality-of-Service (QoS) requirements. Fig. 1 shows the general architecture of the Local Loop Controller. The following subsections present the role and responsibility of each architectural module, together with their interactions.

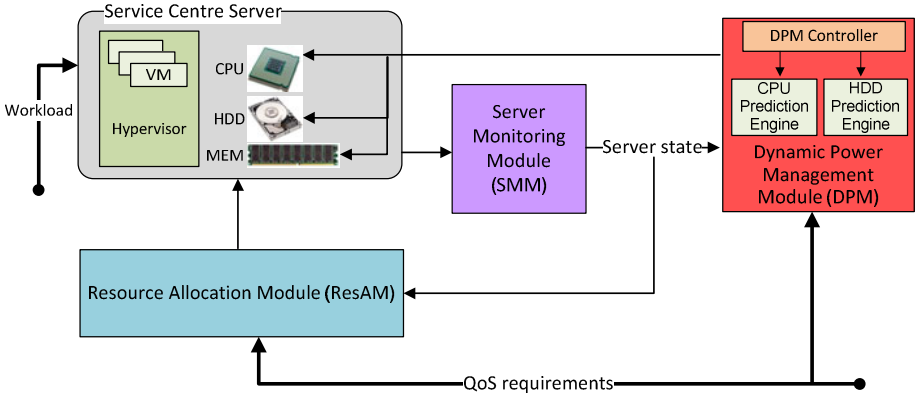


Fig. 1. Local Loop Controller architecture

Resource Allocation Module (ResAM module). The purpose of the ResAM module is to allocate the hardware resources for the virtual machines that are running on the server. The ResAM module considers the performance implications of virtualization and tasks QoS requests, to design a resource allocation strategy which meets performance requests while assuring high energy efficiency.

When dealing with *CPU resource provisioning* the following aspects are considered for a virtual machine: (i) the number of logical processors (i.e. the number of virtual processors to be assigned to the virtual machine); (ii) CPU reservation, representing the minimum percentage of resources that are guaranteed to a specific

virtual machine and (iii) CPU upper limit, representing the maximum percentage of resources that a specific virtual machine is able to use. Another important aspect for the CPU provisioning is to set processor's affinity to a virtual machine, taking into account whether the processor frequency can be scaled per socket or per core. We have defined two CPU resource provisioning strategies: (i) one that tries to distribute as many tasks as possible per core (allowing the unused cores to switch to low power) and per processor (allowing the unused processors to stay in idle as much as possible) and (ii) one that evenly distributes tasks to get a lower overall utilization and thus enable the transition of the entire core/socket to inferior frequencies.

When dealing with the *HDD resource provisioning,* the locality and performance aspects are considered. In a server system having several disk drives used as a cumulative storage space, we distribute the virtual machines as locally as possible. We consider that the virtual machines use a simple file as hard-drive and we can allocate that file on any physical disk for having more drives in spin-down mode as long as possible. This way, all the disks will be accessed, some of them being more active while others having longer idle periods, allowing thus for the dynamic power management controller to spin them down more often and save energy. The major bottleneck in virtual machine performance is the I/O sub-system of the host. Hard disk drives suffer from serious performance degradation when it comes to multi-tasking since most I/O controllers serve request on a first-come, first-serve basis. Therefore, running a large number of virtual machines on a single hard drive, although it has enough storage space, will ultimately result in unacceptable performance penalties. Each virtual disk file has an associated Performance Impact (PI), ranging from 0 to 100. As a result, the cumulative PIs for the virtual disk files stored on a physical drive cannot exceed 100.

The *MEM resource provisioning* is a rather simple process. The virtual machine has a memory requirement and the hypervisor meets it by allocating the specified amount of memory to the virtual machine. The MEM allocation strategy is based on a modified version of the dynamic memory allocation principle. Consider the situation in which we detect that the internal memory allocated to the machine is not enough and that the virtualized guest OS is frequently accessing the swap memory, which in turn translates into hard-drive accesses for the physical host system. If the physical system has unused internal memory, it can supplement the memory size for the virtual machine. This rudimentary caching mechanism will provide the Local Loop Controller with more opportunities to spin-down the hard drive. To reconfigure without powering down the virtual machines, the guest operating system should be able to handle memory as a hot-plug device, i.e. when new memory is added, the gust OS must recognize and use it.

Server Monitoring Module (SMM module). The roles of the SMM module (see Fig. 2) is to continuously analyze the state of the host service center server and of the guest virtual machines and provide the relevant performance and energy consumption data to the ResAM and DPM modules. The ResAM module uses the data about the usage degree of the hardware resources of the host server to decide on an appropriate assignment of virtual machines, while the DPM module uses the data about the host server hardware resources current usage and workload to evaluate the opportunity for a power state transition. For a server, two different aspects are monitored: (i) the

workload generated by the programs which are directly executed on the physical system and monitored by the operating system (such as the operating system, the device drives associated with physical components and the Local Loop Controller itself) and (ii) the server total workload which includes in addition the workload generated by the hosted virtual machines, both in terms of CPU cycles and the amount of data read from / written to the physical hard drives; the virtual machines residing on a server are monitored by means of the virtualization software (hypervisor).

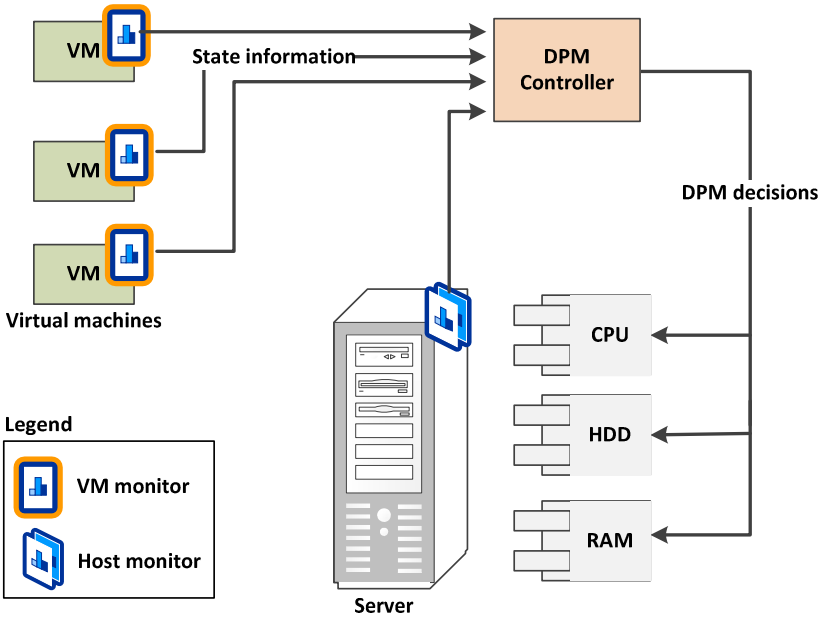


Fig. 2. Server Monitoring Module architecture

Dynamic Power Management (DPM) module. The DPM module uses the data provided by the SMM module, analyzes the possibility of optimizing the server energy efficiency and decides on two types of power management actions: (1) processor power adaptations actions – the processor's frequency is scaled up or down, by means of P-state management, consequently modifying its performance capacity and the amount of energy it requires and (2) hard disk sleep transitions – when the controller detects an idle period longer than the HDD break-even time and decides to put the HDD to sleep state thus reducing its energy consumption. For each of the processor and the hard disk drives, there is an actuator that enforces power management decisions and a prediction engine component that records relevant patterns in workload fluctuations and determines (probable) future idle periods (see Fig. 3). For CPU predictions, a Fuzzy Controller is used, while for HDD a specific idle period prediction engine is used. The power state transition decision for a CPU or HDD must take into account all the virtual machines stored on the host server. A correlation engine analyzes the individual pattern predictions and outputs a global prediction. Regarding the CPU, the correlation engine has to select an appropriate P-state such that the

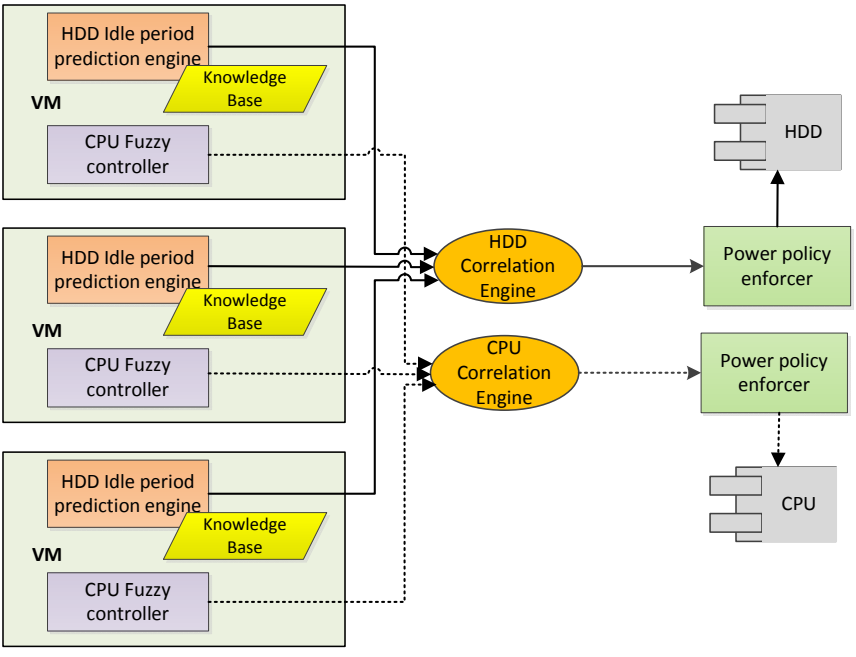


Fig. 3. The DPM Controller

processor-related QoS requirements are satisfied for all the virtual machines under the current workload. A HDD drive can be powered down only if no load is expected in the near future for all virtual machines that use it. As a result, the correlation engine has to check whether all virtual machine associated predictors have issued long idle periods which have not been revoked.

3 Dynamic Power Management (DPM) Strategies

We have defined customized DPM strategies/algorithms for the processor and HDD.

The *power management algorithm for processor* is based on adaptively changing the processor P-states (defining the processor performance levels) taking into account the incoming workload. The algorithm must filter the situations in which the workload fluctuates for short periods of time because the transition cost (in terms consumed energy) can outweigh the benefit of the adaptation. Therefore, if the CPU is currently in a low power state and the load exhibits an isolated spike, the CPU must not enter full mode, because even if there will be a delay in servicing the requests of the load spike, it is tolerable in the context of the overall computational throughput. Given the above considerations, we have decided to design and implement a processor power management algorithm based on fuzzy controller. The approach is somewhat similar to that proposed in [5], however instead of using it to globally manage the service center sizing according to workload, we employ it to control the processor's capacity. The advantage of fuzzy-logic is the ability to filter-out noise and to

adapt progressively to changes. The fuzzy controller has two input variables: the workload represented by the instructions that the processor must execute, and the processor usage ratio. For each of the processor power states, we define three fuzzy sets, LOW, MEDIUM and HIGH, represented by f_l, f_m and f_h functions. After each time window, the fuzzy controller (represented by f_c function) evaluates the f_l, f_m and f_h functions for the current load and updates f_c value according to following equation:

$$f_c = f_c + \propto f_h(load) - \beta f_l(load) + \gamma f_m(load) . \quad (1)$$

When the fuzzy controller reaches the values 0 or 1, the CPU is transitioned to an inferior or a superior power state as appropriate (if there is one available) and its f_c value is reset to the default value of 0.5. By varying α, β and γ, we can control the compromise between energy efficiency and performance degradation: a large α or β value means that the load spikes will not be filtered, while small α and β induce delays that lead to performance degradation and thus increasing energy efficiency.

The *power management algorithm for the hard disk* identifies the hard disk drive access patterns and decides if it is possible to spin-down the drive. Unlike the processor, which is very flexible when it comes to power management, the hard disk state transitions are more rigid and involve significant performance degradation when the power management decision is not adequately performed. For a HDD we identify three distinct power states: (1) *active (Read*, *Write*, *ReadWrite)* - the device is busy, spinning at maximum speed, (2) *standby (Idle)* - the device is not used and it spins at low speed and (3) *sleep* - the device is completely shut down.

The proposed power management algorithm for the HDD, transforms the hard disk access sequences followed by idle periods into discrete quantifiable events and

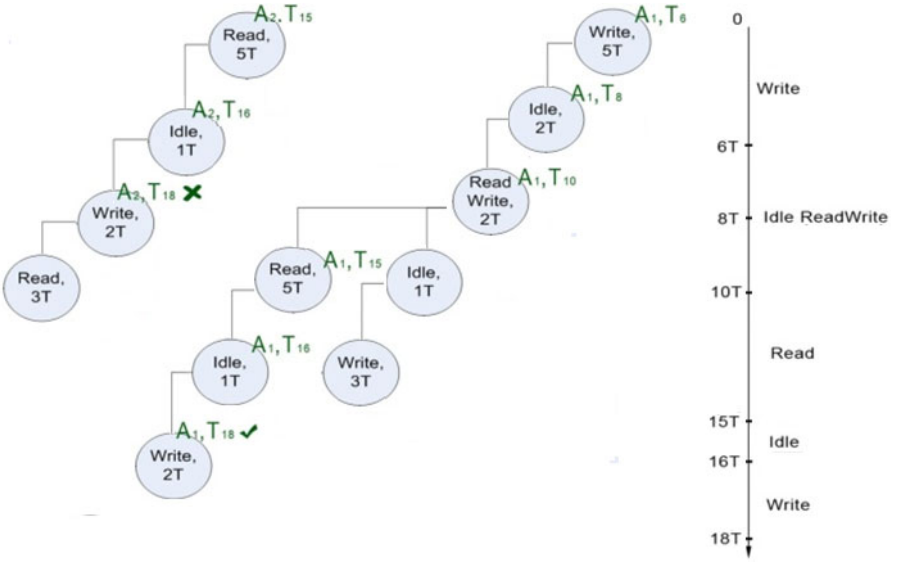


Fig. 4. The development of an adaptive learning tree for identifying HDD idle periods

stores them as nodes in a tree data structure (agent oriented adaptive learning tree [9]). To obtain more accurate results we use an instance of the algorithm for each virtual machine and we additionally define a prediction synchronization phase to determine the opportunity of disk spin down. Every time an event is triggered (state change), a new agent is created in the root of the adaptive learning tree and on every state change the agent will try to advance in the tree. When an agent reaches a leaf, it means that the current sequence found in the adaptive learning tree is supposed to be followed by an idle period. A reward/ penalty approach is used to mark the credibility of the sequence (misprediction – penalty, correct prediction – reward). Fig. 4 shows a possible pattern in the adaptive learning tree knowledge base where T is the length of a monitoring sample period. At t = 6T, a transition is detected (Write->Idle), so an agent (A1) is created that tries to navigate through the learning tree, knowing that the previous state was "Write for 5T". When the next transition is detected (Idle->ReadWrite), A1 advances in the tree, if the current location contains a child annotated with " ReadWrite for 2T". The agent will continue to explore the adaptive learning tree based on state transition events until it reaches a leaf and signals that a known sequence has been detected and the hard drive can be put in *standby* state.

4 Case Study and Results

The test server used to validate the Local Loop Controller power management algorithms is an IBM/PC computer having an Intel Core 2 CPU E6600@2.40GHz (2 cores, 2 logical processors), 2.00 GB Physical Memory (RAM) DDR2@800Mhz and two hard disk drives - 250GB@7200rpm and 320GB@7200rpm. The HDD's operate on the SATA 3 GB/s interface, offering a sustained transfer rate of 126MB/s. The processor provides hardware assisted virtualization functions in the form of Intel Virtualization Technology (Intel-VT) which allows running of Type 1 (native) hypervisors. Two P-states are provided, one at 1.60 GHz (66.67% of standard frequency) and one at 2.40 GHz (100% of standard frequency). The hard disk drives present an active state when they run at full speed, standby states when they spin down and a sleep state when they are completely powered off.

The server underlying operating system is Windows 2008R2 X64. The current implementation use Hyper-V R2 for virtualization support. Hyper-V R2 is a Type 1, hypervisor, providing good performance, flexibility for configuring virtual machines and live migration features. For system monitoring, we use WMI (Windows Management Instrumentation) implemented in the OS and in the hypervisor.

For testing the dynamic power management algorithms, we have defined a test case scenario based on two virtual machines. The first virtual machine (VM1) is configured as having 1GB of RAM and 2 virtual processors. The virtual machine runs an instance of Microsoft SharePoint Server 2007 and an ASP.NET web based application, heavily relying on SQL Server 2005 for data storage. From three computers connected to the same intranet we simulate 15 independent users accessing the application and performing common tasks (such as web pages access, document creating, list management) requiring medium CPU usage and a relatively heavy I/O workload. The second virtual machine (VM2) is configured with 512MB of RAM and one virtual processor and runs a processor intensive application that periodically drives the

CPU to 100% usage level, followed by a lower processor usage. From an I/O perspective we simulate a random light workload.

Since the two virtual machines run on the system, the Local Loop Controller monitors the workload level and takes DPM decisions accordingly. Fig. 5 shows the fuzzy algorithm behavior on a complex workload. As pointed by the green markers in the graph, the algorithm is able to filter random request spikes in the workload of the virtualized applications. The yellow markers show that there is a slight delay between the workload fluctuation and the P-state transition.

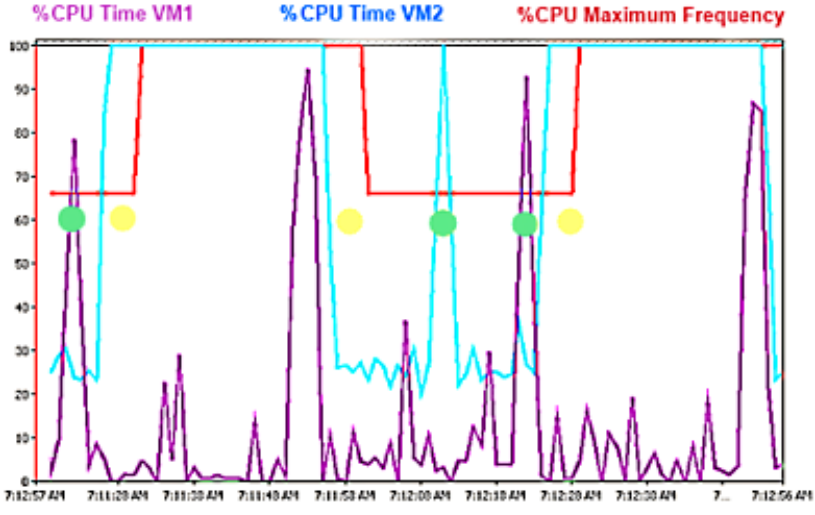


Fig. 5. Results of fuzzy logic based power management algorithm for processor

To fully evaluate the advantages brought by the proposed power management algorithms, the processor P-state transition results were compared to the results obtained by using the Balanced Power Scheme implemented in Windows Server 2008R2. As illustrated in Fig. 6, the Balanced Power Scheme processor transitions are more aggressive, and even isolated request spikes determine P-state changes. This shows that, the operating system's emphasis is on performance and not on energy efficiency and also that the power management algorithm is fairly rudimentary. Using our processor power management algorithm, the processor spends 39% of the time in low power state, while using the Balanced Power Scheme it is at low frequency for 23% of time (for the same workload).

Regarding the Balanced Power Scheme for HDD, Windows never spins down the drives; it does allow setting a time-out, but not less than 1 minute. Considering a light I/O workload, during a one hour test using the Balanced Power Scheme and with the spin-down time-out set at the minimum allowed of 1 minute, the hard drive was in idle for 312 seconds (~5.2 minutes). On the other hand using the same test conditions, the Local Loop Controller has transitioned the drive to idle for a total 1447 seconds (~24.5 minutes), resulting an increase of the hard disk idle period with 470%.

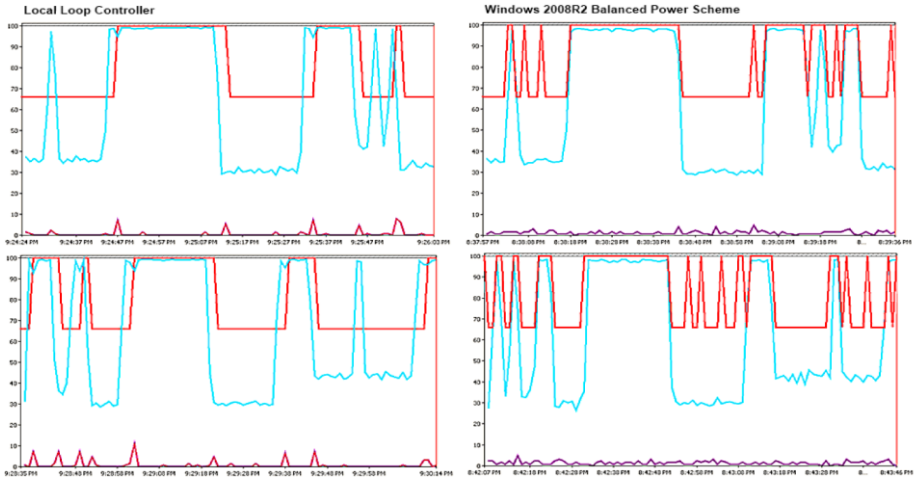


Fig. 6. P-state transition results: Local Loop Controller vs. Windows Balanced Power Scheme

Considering the hardware configuration of the test system, the hardware manufacturers technical data and the processor/hard disks idle periods we can estimate the energy consumption for the system at different load levels. In our test conditions, we estimate an energy consumption of 245Wh (watt hour) for the Balanced Power Scheme and of 230 Wh for our Local Loop Controller. This results in an estimated energy savings of around 15 Wh. Using the time to execute the workload as performance factor we estimate a 4% performance degradation when our Local Loop Controller algorithms are used, compared to the OS default power scheme.

5 Conclusions

This paper proposes the development of a management controller, which balances the service center servers' workload and hardware resource usage to locally optimize the servers' energy consumption. The results are promising showing that using the proposed dynamic power management strategies the hard disk idle periods increase with approximately 470% compared with Windows Balanced Power Scheme. Also an improvement of 16% of the processor low power state periods is obtained, when using the Local Loop Controller instead of the Balanced Power Scheme.

Acknowledgments

This work has been supported by the European Commission within the GAMES project [2] funded under EU FP7.

References

1. U.S. Environmental Protection Agency, ENERGY STAR Program, Report to Congress on Server and Data Center Energy Efficiency, Public Law 109-431 (2007)
2. GAMES Research Project, `http://www.green-datacenters.eu/`

3. Khargharia, B., Hariri, S., Yousif, M.: Autonomic power and performance management for computing systems. Cluster Computing 11(2), 167–181 (2008) ISSN:1386-7857
4. Wang, N., Kandasamy, N., Guez, A.: Distributed Cooperative Control for Adaptive Performance Management. IEEE Internet Computing 11(1), 31–39 (2007)
5. Jeske, J.C., Julia, S., Valette, R.: Fuzzy Continuous Resource Allocation Mechanisms in Workflow Management Systems. In: IEEE International Conference on Information Reuse and Integration, pp. 472–477 (2006) ISBN: 0-7803-9788-6
6. Minerick, R., Freeh, V., Kogge, P.: Dynamic Power Management using Feedback. In: Proceedings of Workshop on Compilers and Operating Systems for Low Power (2002)
7. Gupta, R., Irani, S., Shukla, S.: Formal Methods for Dynamic Power Management. In: IEEE/ACM International Conference on Computer-Aided Design, p. 874 (2003)
8. Bircher, L., John, L.: Analysis of Dynamic Power Management on Multi-CoreProcessors. In: Proc. of the 22nd Annual International Conference on Supercomputing, pp. 327–338 (2008)
9. Chung, E., Benini, L., De Micheli, G.: Dynamic Power Management Using Adaptive Learning Tree. In: IEEE/ACM Int. Conf. on Computer-Aided Design, pp. 274–279 (1999)
10. Bisson, T., Brandt, S., Long, D.: NVCache: Increasing the effectiveness of disk spin-down algorithms with caching. In: Proceedings of the 14th IEEE International Symposium on Modeling, Analysis, and Simulation, pp. 422–432 (2006)
11. Zhu, Q., David, F., Devaraj, C., et al.: Reducing Energy Consumption of Disk Storage Using Power-Aware Cache Management. In: Proceedings of the 10th International Symposium on High Performance Computer Architecture, p. 118 (2004)
12. Stoess, J., Lang, C., Bellosa, L.: Energy Management for Hypervisor-Based Virtual Machines. In: USENIX Annual Technical Conference (2007)
13. Kansal, A., Zhao, F., Liu, J., et al.: Virtual Machine Power Metering and Provisioning. In: The 1st ACM Symposium on Cloud Computing, pp. 39–50 (2010) ISBN:978-1-4503-0036-0
14. Colarelli, D., Grunwald, D.: Massive arrays of idle disks for storage archives. In: Proceedings of the ACM/IEEE Conference on Supercomputing, pp. 1–11 (2002)

An Energy Aware Context Model for Green IT Service Centers

Ioan Salomie[1], Tudor Cioara[1], Ionut Anghel[1],
Daniel Moldovan[1], Georgiana Copil[1], and Pierluigi Plebani[2]

[1] Technical University of Cluj-Napoca, Computer Science Department,
Cluj-Napoca, Romania
{Ioan.Salomie,Tudor.Cioara,Ionut.Anghel}cs.utcluj.ro
[2] Politecnico di Milano, Computer Engineering,
Milano, Italy
Plebani@elet.polimi.it

Abstract. In this paper we propose the development of an Energy Aware Context Model for representing the service centre energy/performance related data in a uniform and machine interpretable manner. The model is instantiated at run-time with the service center energy/performance data collected by monitoring tools. Energy awareness is achieved by using reasoning processes on the model instance ontology representation to determine if the service center Green and Key Performance Indicators (GPIs/KPIs) are fulfilled in the current context. If the predefined GPIs/KPIs are not fulfilled, the model is used as primary resource to generate run-time adaptation plans that should be executed to increase the service center's greenness level.

Keywords: Service Center, Context Model, Energy Awareness, Reinforcement Learning.

1 Introduction and Related Work

Over the last years the energy efficiency management of IT processes, systems and service centers has emerged as one of the most critical environmental challenges to be dealt with. Since computing demand and energy costs are continuously growing, energy consumption of IT systems and service centers is expected to become a priority in the future years [1].

The GAMES (Green Active Management of Energy in IT Service Centers) EU FP7 research project [2] aims at developing a set of innovative methodologies, metrics, services and tools for the active management of energy efficiency of IT service centers. The GAMES vision is to create a new generation of Green and Energy Aware IT service centers by defining and implementing management actions in both design time and run-time for increasing energy efficiency. The problem of service centre run-time energy efficiency in the GAMES project is approached by dynamically finding and executing Dynamic Power Management (DPM) and Consolidation based adaptation actions targeting the identification of the over provisioned resources with the goal of putting them in low power states. To determine run-time adaptation decisions, the

E.M. Maximilien et al. (Eds): ICSOC 2010 Workshops, LNCS 6568, pp. 169–180, 2011.

following MAPE (Monitoring, Analysis, Planning and Execution) steps are taken: (i) the current service center energy/performance data is captured using monitoring tools, (ii) using the collected data, the current values for GPIs (Green Performance Indicators) and KPIs (Key Performance Indicators) are evaluated and compared against their predefined values (iii) if the GPIs and KPIs are fulfilled no action is taken; otherwise a plan of adaptation actions is determined and executed to enforce the GPIs/KPIs predefined values.

The service center energy/performance data collected from various sources (such as sensors, monitoring devices, software applications, etc.) is represented in heterogeneous formats that are difficult to interpret and analyze. There is an evident need for providing an integrated, uniform and semantically enhanced energy/performance data representation model that can be automatically processed and interpreted at run-time. To solve these problems we have developed an Energy Aware Context Model (EACM). The proposed EACM model is constructed by mapping our RAP (Resources, Actions and Policies sets) generic context model [16] onto the service centre energy efficiency domain. An EACM instance representing a service centre snapshot is created by instantiating at run-time the EACM model elements with the service center energy/performance data collected using monitoring tools. To ensure the energy awareness, the EACM model instance, implemented as ontology is analyzed/processed at run-time by using reasoning processes, to determine if the GPIs/KPIs are fulfilled for the current service center snapshot and to generate/execute adaptation action plans if these indicators are not fulfilled.

The state of the art literature contains many references regarding context modeling, but none of them (as of our knowledge) approaches the energy efficiency problem by considering the energy consumption as a relevant context feature. Also, there are no approaches for context modeling of Green IT service centers.

The most important problems regarding context information acquisition refer to identifying the features of the system execution context [3], [20], and defining models for capturing features' specific data [4]. In the domain literature ([5], [6] and [7]), several characteristics that may define the context are considered, such as spatiotemporal (time and location), ambient, facility (the system devices and their capabilities), system user interaction, system internal events, system life cycle, etc. Our paper takes this work one step further and introduces the energy consumption and the resource usage as an important characteristic of the modeled context.

Regarding the context representation, generic models that aim at accurately describing the context in a programmatic manner are proposed [19]. In [8] the use of key-value models to represent the set of context features and their associated values is proposed. Markup models [9] and object oriented models [10] are also proposed to structure and represent the context information. The main disadvantage of these approaches is their high degree of inflexibility and lack of semantics. Alternatively, the use of ontologies to model the context data, where context features are represented as ontological concepts and instantiated with run-time captured values are more and more used [11]. We took advantage of the semantic-oriented and reasoning features of the ontologies and developed an ontology based representation of the EACM model.

For context analyzing, models and techniques aiming at determining and evaluating the context changes are proposed. These models are strongly correlated with the context representation model. In [12] fuzzy Petri nets are used to describe context

changing rules. Context analyzing models based on reasoning and learning about context information are proposed in [13], [14] and [15] where context change rules are described using natural language or first order logic and evaluated using reasoning engines. Our paper uses reasoning algorithms to evaluate and analyze the run-time changes targeting the energy awareness.

The rest of this paper is structured as follows: Section 2 presents the EACM model, Section 3 shows how energy awareness is enacted, Section 4 describes a case study and evaluation results, while Section 5 concludes the paper.

2 EACM – The Energy Aware Context Model

This section introduces the EACM model highlighting the main concepts and relations defined and used to represent the service center energy / performance related data. The EACM model is constructed by mapping the RAP context model [16] onto the service centre energy efficiency domain. In the RAP model the context data is represented as triple <R, A, P> where R is the set of context resources, A is the set of context adaptation actions and P is the set of context policies. *Context resources (R)* represent the physical or virtual entities that generate and / or process context data. *Context actions (A)* represent a set of possible adaptation actions that have to be executed to enforce a predefined set of conditions for a given context situation. *Context policies (P)* are used to define a set of rules that must hold in the context, being used for controlling the interactions within the context. Fig. 1. shows the service centre energy efficiency domain specific concepts (highlighted in blue) and their classification into the RAP context model main sets (highlighted in red).

Context Resources (R). Three types of context resources that generate/collect context data were identified in service centers: (1) Service Centre IT Facility Context Resources (Facility Resource for short), (2) Service Centre IT Computing Context Resources (Computing Resource for short) and (3) Business Context Resources (Business Resource for short). *Facility Resources* are physical or virtual entities which provide or enforce the service centre ambient properties. A Facility Resource is characterized by the ambient property data type that resource can capture or modify. Passive Resources capture and store service centre ambient data, while Active Resources execute adaptation actions to modify the service centre ambient properties. For example, a temperature value can be the property for a temperature sensor resource (non-modifiable property) as well as for an air cooling resource (this time a modifiable property). *Computing Resources* are physical or virtual entities which supply context data related to the actual workload and performance capabilities of the service center. A Computing Resource can be also defined as a resource which consumes energy as a result of executing a specific workload. In our model we are interested to represent only those service centre computing resources that allow executing of Dynamic Power Management (DPM) actions aiming at setting a computing resource into different power states, according to its current workload. A Computing Resource is characterized by the list of energy consuming states property. For example, an Intel Core i7 860@2.8Ghz processor has 12 P-states varying from 1197Mhz (40%) up to 2926Mhz (100%). The Computing Resources are classified as Simple and Complex (see Fig. 1). A Simple

Computing Resource provides only one atomic performance property throughout its lifecycle. For example, CPU energy-related performance is characterized only by its frequency value. A Complex Computing Resource is composed from a set of Simple Resources and is characterized by a set of performance properties. For example the energy-related performance of a server is expressed by means of its component's energy-related performance properties such as the spindle speed for HDD or the clock rate for CPU. A *Business Resource* is a virtual entity which provides information about the QoS requirements of the executed application.

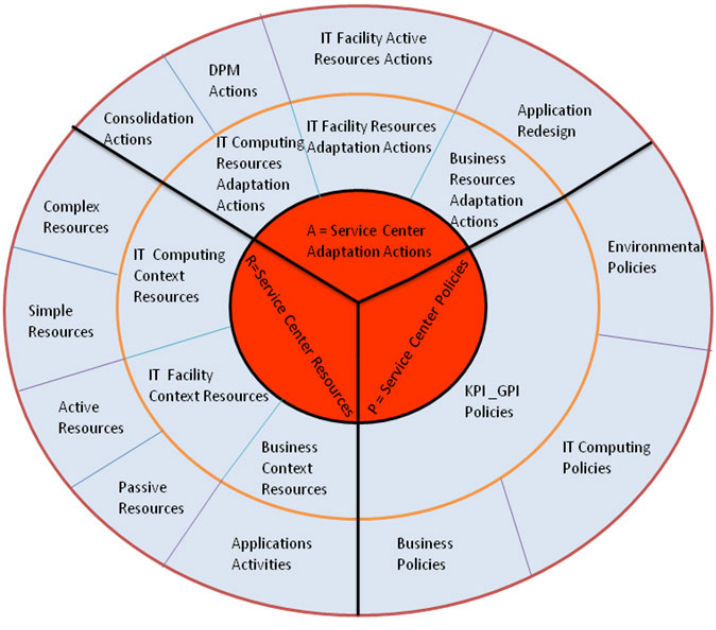


Fig. 1. EACM model elements obtained by mapping RAP model onto service center energy efficiency domain

Context Actions (A). Three types of service centre adaptation actions are identified: (1) IT Computing Resources Adaptation Actions, (2) IT Facility Resources Adaptation Actions and (3) Application Adaptation Actions. *IT Facility Resources Adaptation Actions* (e.g. adjust the room temperature or start the CRAC) are enforced through the Active Resources. *IT Computing Resources Adaptation Actions* are executed to enforce the set of predefined GPI and KPI indicators on the Computing Resources. We have defined two types of IT Computing Resources Adaptation Actions: Consolidation Actions and DPM Actions. Consolidation Actions aim at identifying the Computing Resources for which the workload is inefficiently distributed from the energy efficiency point of view and balancing workload distribution in an energy efficient manner. DPM Actions aim at determining the over provisioned resources with the goal of putting them into low power states. *Application Adaptation*

Actions should be executed during design-time on the service centre applications activities (such as application redesign for energy efficiency).

Context Policies (P). The constraints regarding the service centre energy / performance are modeled through a predefined set of GPI and KPI related policies. We have identified and modeled three categories of GPI and KPI policies (see Fig. 1): (1) Environmental Policies, imposing restrictions about the service centre ambient conditions, (2) IT Computing Policies, describing the energy/performance characteristics of the service centre that and (3) Business Policies, describing the rules imposed by the custom business for the application execution. The Context Policies are described using the XML based policy description model proposed by us in [18].

Energy Aware Context Model Elements Relations. To model the interactions between the EACM model elements we have defined three types of relations: (1) proper subset relations, (2) trans-set 1 to N relations and (3) trans-set 1 to 1 relations. The *proper subset relations* reflect the hierarchical relations between the EACM model elements. For example, the Context Resources set is populated with service centre resources classified in three main proper subsets: Computing Resources, Facility Resources and Business Resources. The *trans-set 1 to N relations* model the interactions between an element of an ECAM model set and a subset of elements of the model. For example, this type of relation is used to model and represent the interactions between the energy aware run-time adaptation action and the service center resources on which it is enforced. The *trans-set 1 to 1 relations* model the interactions between two EACM model elements part of different sets. For example, in our model a policy (part of the Context Policy set) may have attached a default adaptation action (part of the IT Computing Adaptation Actions set) that has to be executed when the policy is broken.

3 Enacting Energy Awareness

To assure energy awareness, the service center context situation (snapshot) is represented in a programmatic manner using the EACM model instance ontology implementation (see Fig. 2). The EACM model instance is processed and interpreted at run-time by means of reasoning to: (1) evaluate if the GPIs/KPIs context policies are fulfilled for the current service centre context situation and (2) generate/execute adaptation action plans if the GPIs/KPIs policies are not fulfilled.

To *evaluate the GPI/KPI policies,* reasoning rules are used. The policies are converted into reasoning rules and automatically evaluated (without human intervention) by means of a reasoning engine, using the EACM model instance ontology representation. Fig. 3 shows the evaluation of GPIs/KPIs context policies using as an example a policy describing the accepted performance/workload values for a server and its SWRL (Semantic Web Rule Language) rules representation.

To measure the degree of fulfilling the set of GPIs/KPIs related policies we have defined the concept of service centre context situation entropy (E_S) and its associated

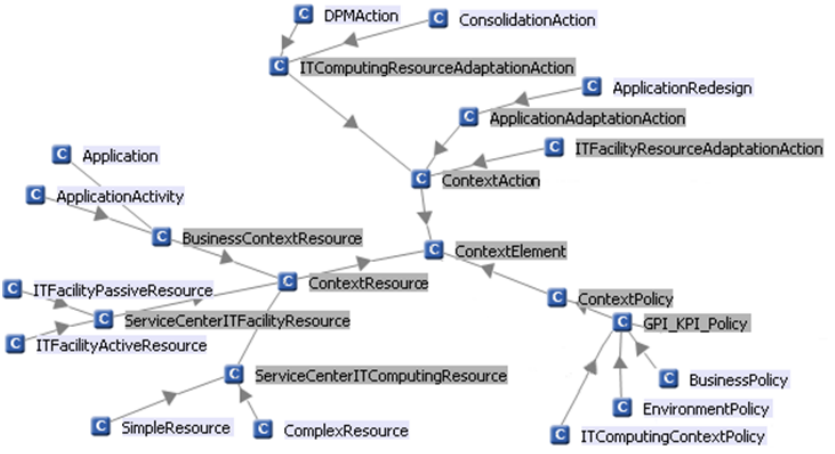


Fig. 2. EACM model elements implemented as ontological classes

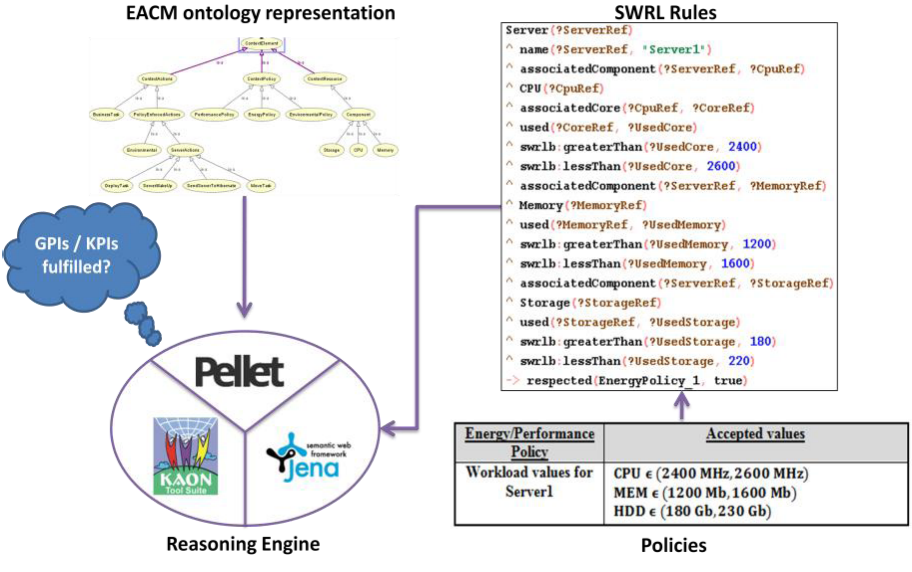


Fig. 3. The GPI/KPI policy evaluation

threshold (T_E). The entropy is an indicator that measures the level of compliance to the service center specific energy-saving requirements (i.e. the greenness level) [21]. If the evaluated context situation entropy is below a predefined threshold T_E, then all the GPIs/KPIs policies are fulfilled and adaptation is not required. Otherwise, adaptation actions must be executed to enforce the broken GPIs/KPIs policies and to bring the entropy below T_E. The EACM model instance entropy is computed as in relation (1) where: (i) pw_i is the weight of the GPI/KPI policy i and represents the importance of the policy in the service centre context, (ii) rw_{ij} is the weight of the service centre

context resource i in the policy j and reflects the service centre resource importance for that policy and (iii) v_{ij} is the deviation between the recorded value for service centre resource j and the accepted value defined by policy i.

$$E_S = \sum pw_i \sum rw_{ij} * v_{ij} \quad (1)$$

Taking into account the toleration to changes, we have defined two types of entropy thresholds: *a restrictive threshold* and *a relaxed threshold.* For the first case, we define the threshold at the lowest possible entropy value ($T_E = 0$). Whenever a GPI/KPI policy imposed restriction is broken, the adaptation process is triggered. In the second case, for each GPI/KPI policy we define an accepted entropy contribution value E_i and compute the entropy threshold T_E (relation 2). The broken GPIs/KPIs policies are tolerated if their entropy contribution is lower than the accepted value.

$$T_E = \sum pw_i * v_{ij} * (100 + E_i \,\%\, 100) \quad \text{where} \quad E_i = pw_i \sum ri_{ij} * v_{ij} \quad (2)$$

To *generate and execute adaptation action plans,* we identify first the previously encountered similar service center context situations in which the same GPIs/KPIs context policies were broken. If such a similar equivalent situation is found, the same action plan is selected and executed (see Fig 4).

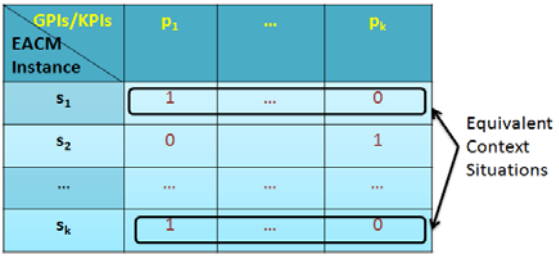


Fig. 4. Equivalent service center context situations

Otherwise, a new sequence of adaptation actions is generated using a reinforcement learning approach (what / if analysis). The learning process considers all possible service center context situations (represented as EACM instances) and builds a decision tree by simulating the execution of all available adaptation actions for each situation (see Fig 5). Each tree node stores a service center context situation and its calculated entropy value. A tree path between two nodes S1 and S2 defines a sequence of adaptation actions which, executed in S1 context situation, generates the new service center context situation stored in node S2. The minimum entropy path in the reinforcement learning decision tree represents the best sequence of adaptation actions that when executed, will bring the service center in a context state in which all GPIs/KPIs context policies are fulfilled.

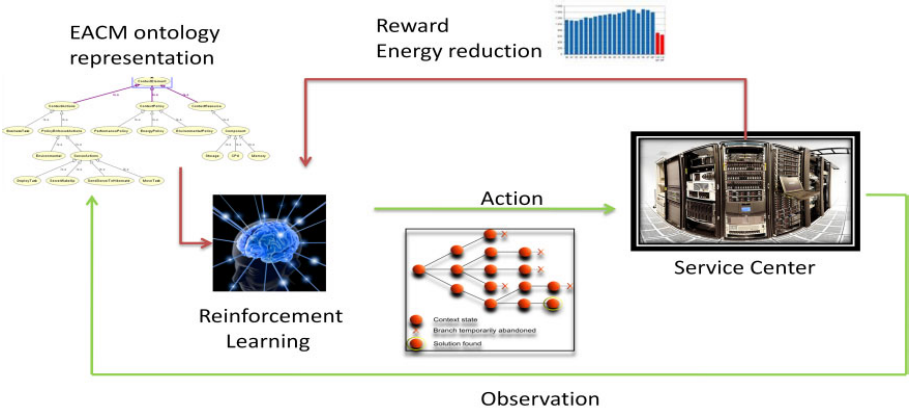


Fig. 5. Reinforcement learning based adaptation action plans generation

The learning process may generate different type of results discussed below.

Case 1: The algorithm finds only a possible service center context situation with an entropy value lower than the defined threshold. The sequence of actions that lead to this context situation is selected and the search process is stopped.

Case 2: The current service center context situation entropy is higher than the threshold, but smaller than the minimum entropy determined so far. The minimum entropy is replaced with the new entropy and the search process is continued.

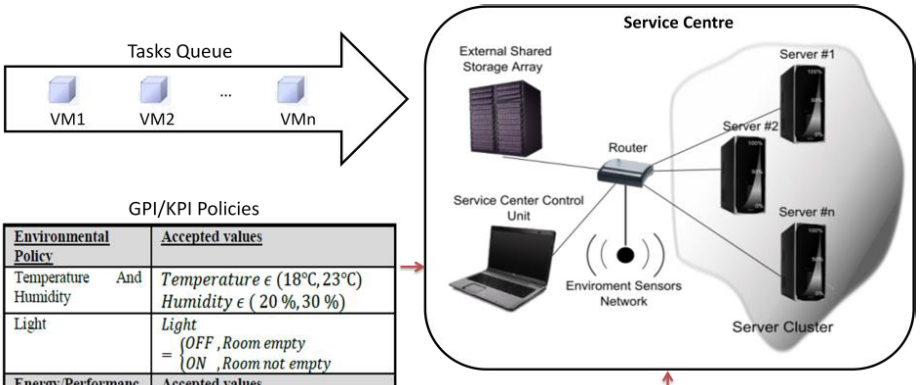
Case 3: The current entropy is higher than both the threshold and the minimum entropy; the reinforcement learning algorithm continues the search process. If at a certain moment, all exercised paths of the decision tree are cycles, the algorithm stops and chooses the path leading to a state with the minimum entropy.

4 Case Study

In this section the *energy awareness capabilities* of the proposed EACM model and *the scalability* of its ontology representation are discussed and evaluated.

To **evaluate EACM model energy awareness capabilities,** we used it to manage a small service center with the following configuration: (i) a server cluster composed from three physical servers on which virtualized applications, annotated with Quality-of-Service (QoS) requirements, are executed, (ii) an external shared storage server and (iii) a set of sensors and facilities interconnected through a sensor network which control the service centre environment. Fig. 6 presents the test case service centre infrastructure together with the set of defined GPIs/KPIs policies.

To determine and analyze the time necessary for *evaluating the GPIs/KPIs context policies*, we have generated two categories of service center context situations: (i) situations in which the number of GPIs/KPIs broken policies is gradually increasing (each broken policy having similar complexity) and (ii) situations in which GPIs/KPIs with increasing complexity are broken. The complexity is measured in terms of the number of atoms in the antecedent of the policy corresponding SWRL rule. For the first context situations category, the test started from three broken GPIs/KPIs policies



GPI/KPI Policies

Environmental Policy	Accepted values
Temperature And Humidity	$Temperature \in (18°C, 23°C)$ $Humidity \in (20\,\%, 30\,\%)$
Light	$Light = \begin{cases} OFF, Room\ empty \\ ON, Room\ not\ empty \end{cases}$
Energy/Performance Policy	**Accepted values**
Workload values for Server 1	$CPU \in (600\,MHz, 2065\,MHz)$ $MEM \in (1000\,Mb, 4000\,Mb)$
Workload values for Server 2	$CPU \in (1000\,MHz, 2465\,MHz)$ $MEM \in (1750\,Mb, 4750\,Mb)$

Computing Resources Configuration

Name	Cores No.	CPU Freq.	Memory	Storage	Power source
Server1	8	2930 MHz	6144 MB	-	500 W
Server2	8	2930 MHz	6144 MB	-	500 W
Server3	8	2930 MHz	6144 MB	-	500 W
Storage Server	2	2400 MHz	2048 MB	600 GB	350 W

Fig. 6. Test case service centre infrastructure

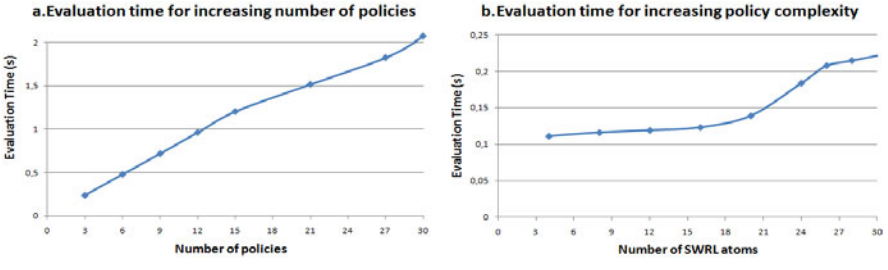


Fig. 7. The EACM model GPIs/KPIs context policies evaluation results

and continued by gradually increasing this number up to 30 broken policies. For the second category of context situations, starting from one broken GPI/KPI policy with 4 atoms the number of atoms was grown up to 34. The results show (Fig. 7b) that for less than 20 atoms the evaluation time is within reasonable limits (less than 0.15sec.). When the complexity of the policy increases above 20 atoms the evaluation time grows exponentially. Reasonable time results (Fig. 7a) were also obtained for evaluating up to 30 context policies at once (about 2 sec.).

To determine the time needed for the *generation of the adaptation action sequence* when some GPIs/KPIs are broken, the EACM model was used to manage randomly generated service center context situations by means of reinforcement learning for about 27 hours (see Fig. 8). In the first 1000 decaseconds (das), almost all running times of the adaptation action selection algorithm are greater than 10 seconds. After

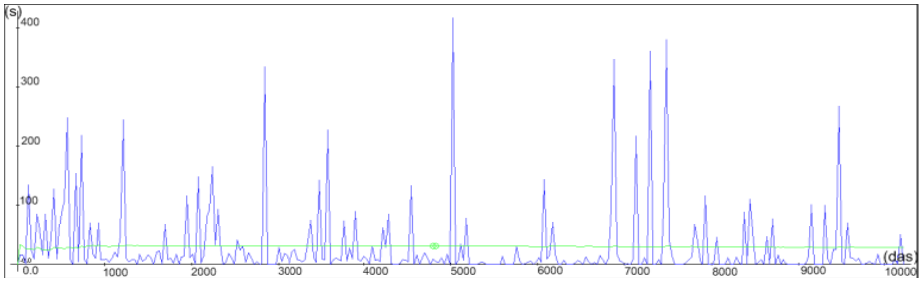


Fig. 8. The EACM model adaptation action sequence generation

that, the reinforcement learning mechanism begins to learn and achieving as a result the performance of having only four running times (the spikes in Fig. 8) greater than 10 seconds in the [5000, 7000] das time interval. Also, an overall reduction in the height of the peaks is visible because at each step the algorithm tries to determine the equivalent service center context situations and if equivalence is found the same sequence of actions is taken for execution.

To **evaluate the ontology representation of the EACM model,** three criteria were used: (i) instances creation time, (ii) instances retrieval time and (iii) memory usage. Our goal is to determine if the EACM model ontology implementation is feasible/scalable taking into account the above presented criteria and to identify the most suitable tool/API for the EACM ontology management. For ontology management, two strategies were used: (1) the ontology was created and persisted in the RAM memory (using Protégé and Kaon2 Ontology tools) and (2) the ontology was mapped onto a database model persisted in RAM memory. For the second strategy, three database models have been tested: HSQL relational model, Prevayler hierarchical model and Kaon 2 Database relational model.

To estimate the number of EACM model concepts instances that are created for representing different service centers, the following classification was used [17]: (i) small service centers having a number of servers between 101 and 500, (ii) medium service centers having between 501 and 5000 servers and (iii) large service centers with a number of servers over 5000. We consider that an average service center server has four processors each with two cores, four memory slots each of 2Gb and an array of four hard disks each with a capacity of 1Tb. For this type of server the EACM model instance ontology has to store approximately 17 concept instances: one IT Computing Complex Resource (corresponding to the server itself) and 16 IT Computing Simple Resource (corresponding to the server components). The number of instances corresponding to the service center IT infrastructure and the business tasks description are relatively low compared to IT Computing Resource instances, hence they can be ignored for the scalability tests. We have defined an EACM Model management scenario in which a number of instances that have been automatically generated need to be administrated. The EACM model instance generation module takes as input an integer number n and automatically generates n^3 EACM instances for different ontology concepts. By choosing different values for n we have created a number of EACM instances which correspond to each service center type as follows: for n = 15 a number of 3,375 instances are generated corresponding to small service

Metric	Instance Creation Time (min:sec)				Time to retrieve all instances (min:sec)				Memory Usage (MB)			
Instance Count	15^3	25^3	50^3	70^3	15^3	25^3	50^3	70^3	15^3	25^3	50^3	70^3
Protégé	0:05	0:08	0:41	1:15	≈0:00	≈0:00	0:40	2:20	37	148	500	2577
Kaon2 Ontology	0:02	0:06	0:39	1:01	≈0:00	≈0:00	≈0:00	≈0:00	120	211	577	2750
HSQL + Hibernate	0:01	0:03	0:25	01:08	≈0:00	0:04	01:31	7:34	45	56	156	361
Prevayler	0:00	0:01	0:08	0:18	≈0:00	≈0:00	≈0:00	≈0:00	65	123	214	364
Kaon2 Database	0:03	0:07	0:22	01:00	≈0:00	≈0:00	≈0:00	≈0:00	70	110	409	634

Fig. 9. EACM ontology implementation management evaluation results

centers; for n = 25 a number of 15,625 instances are generated, corresponding to medium service centers, while for n = 50 and n = 70 result 125,000 and respectively 343,000 instances corresponding to large service centers.

The results presented in Fig. 9 show that: (i) the instance creation time and the memory usage grow exponentially for a number of instances higher than 50^3 but it remains within acceptable boundaries even for 70^3 instances and (ii) the instances retrieval time can be neglected for Prevayler, Kaon2 Ontology and Database.

5 Conclusions

This paper introduces the EACM model for representing the service centre energy related data in a uniform and machine interpretable manner. Reasoning based processes are defined and used on the model ontology representation to ensure energy awareness. The EACM model evaluation results are promising showing that the energy awareness capabilities can be enacted at a service center level in reasonable time frames using: (i) the model entropy to determine the service center greenness level and (ii) reinforcement learning to determine the adaptation actions to be executed when the defined greenness levels are not reached.

Acknowledgments

This work has been supported by the European Commission within the GAMES project [2] funded under EU FP7.

References

1. Kaplan, J.M., Forrest, W., Kindler, N.: Revoluzioning Data Center Energy Efficiency, McKinsey&Company, Technical Report (2008)
2. GAMES Research Project, http://www.green-datacenters.eu/
3. Wang, K.: Context awareness and adaptation in mobile learning. In: 2nd IEEE Int. Work. on Wireless and Mobile Tech. in Education, pp. 154–158 (2004) ISBN:0-7695-1989-X

4. Yu, Z.: iMuseum: A scalable context-aware intelligent museum system. Computer Communications 31(18), 4376–4382 (2008)
5. Burghardt, C., Reisse, C.: Implementing scenarios in a Smart Learning Environment. In: 6th Annual IEEE Int. Conf. on Perv. Comp. and Comm. (2008) ISBN: 0-7695-3113-X
6. Pareschi, L., Riboni, D.: Composition and Generalization of Context Datafor Privacy Preservation. In: 6th Annual IEEE Int. Conf. on Perv. Comp. and Comm. (2008)
7. Grossniklauss, M.: Context Aware Data Management, 1st edn. VDM Verlag (2007) ISBN 978-3-8364-2938-2
8. Anderson, K.M., Hansen, F.A., Bouvin, N.: Templates and queries in contextual hypermedia. In: Proc. of the 17th Conf. on Hypertext and hypermedia, pp. 99–110 (2006)
9. Raz, D., Juhola, A.T.: Fast and Efficient Context-Aware Services. Wiley Series on Comm. Networking & Distributed Systems, pp. 5–25 (2006) ISBN-13: 978-0470016688
10. Hofer, T., Schwinger, W.: Context-awareness on mobile devices – the hydrogen approach. In: The 36th Annual Hawaii International Conference on System Sciences, USA, p. 292 (2003)
11. Cafezeiro, I., Hermann, E.: Ontology and Context. In: 6th Annual IEEE Int. Conf. on Perv. Comp. and Comm. (2008) ISBN: 0-7695-3113-X
12. Huaifeng, Q.: Integrating Context Aware with Sensornet. In: Proc. of 1st Int Conf. on Semantics, Knowledge, Grid (2006) ISBN:0-7695-2534-2
13. Sirin, E., Parsia, B.: Pellet: A practical OWL-DL reasoner, Web Semantics: Science. Services and Agents on the World Wide Web 5(2), 51–53 (2007)
14. Amoui, M., Salehie, M.: Adaptive Action Selection in Autonomic Software Using Reinforcement Learning. In: ICAC 2008, pp. 175–181 (2008) ISBN 0-7695-3093-1
15. Bernstein, A., Kaufmann, E.: Querying the Semantic Web with Ginseng:A Guided Input Natural Language Search Engine. In: 15th Work. on Information Tech. and Syst. (2005)
16. Cioara, T., Anghel, I., Salomie, I.: A Generic Context Model Enhanced with Self-configuring Features. JDIM 7(3), 159–165 (2009) ISSN 0972-7272
17. Material Stock in German Data Centres (2010), http://www.uba-green-it.de
18. Cioara, T., Anghel, I., Salomie, I.: A Policy-based Context Aware Self-Management Model. In: SYNASC 2009, pp. 333–341 (2009) ISBN: 978-0-7695-3964-5
19. Baldauf, M., Dustdar, S., Rosenberg, F.: A Survey on Context Aware Systems. International Journal of Ad Hoc and Ubiquitous Computing 2(4), 263–277 (2007)
20. Truong, H.-L., Dustdar, S.: A Survey on Context-aware Web Service Systems. International Journal of Web Information Systems 5(1), 5–31 (2009)
21. Cioara, T., Anghel, I., Salomie, I., Pernici, B.: A Context Aware Self-Adapting Algorithm for Managing the Energy Efficiency of IT Service Centers. In: UbiCC 2010 (2010)

Creating Environmental Awareness in Service Oriented Software Engineering

Patricia Lago[1] and Toon Jansen[2]

[1] VU University Amsterdam, The Netherlands
patricia@cs.vu.nl
[2] Het Expertise Centrum, The Netherlands
T.Jansen@hec.nl

Abstract. Carbon emission of IT is an issue. ICT energy consumption is expected to grow by 73% (instead of the originally targeted 26%) until 2020, and the service sector alone counts for 70% of the European economy. Energy consumption is a combination of what we use, and how we use it. Most green initiatives look at what types of devices do consume energy, and try to optimize their up-time as such. Few initiatives, though, measure how do software systems actually use these devices, with the goal of optimizing consumption of devices and computing resources. Basic research is needed to address this software optimization problem. The proposed approach is to make visible the environmental impact of software services by measuring it. In this way, we will become aware of the amount of energy needed by our software, and hence learn how to target software optimization where it is mostly needed. As a first step in this direction, in this paper we define three main problem areas to realize green service-based applications, and propose a service-oriented approach to address them. Thanks to that we can bring clarity in what entails managing and developing green software according to environmental strategies.

Keywords: energy-efficient software, green IT, service orientation, green metrics, sustainability.

1 Introduction

In the last decade, Service Oriented Software Engineering (SOSE) and Service-Oriented Computing have emerged as a major software development discipline resembling (if not outbalancing) what object orientation meant for the IT market of the eighties. Service Oriented Architecture (SOA) captures a logical way of providing software services both within an enterprise, and across organizational and national boundaries to either end-user applications or other services distributed on a network [3]. A well-constructed SOA can empower a business or social environment with a flexible infrastructure and processing environment by providing independent, reusable automated business processes (as services), and a robust foundation for leveraging these services [2, 6].

Due to its fast application in all aspects of our society, SOSE should address environmental issues promptly: in this period of acceptation and learning, we should think

E.M. Maximilien et al. (Eds): ICSOC 2010 Workshops, LNCS 6568, pp. 181–186, 2011.

about how to sensitize people to the adoption of an ecological approach to the construction, deployment and use of service-based applications (SBAs). Unfortunately, service-oriented software is often developed assuming availability of unlimited resources: 24x7 service availability, unlimited use of computing power, hardware, network and printing resources (all demanding energy and causing carbon emission). The increase in the network traffic and in the number of electronic data centers is having a huge impact on the environment (an average data center is consuming the same amount of energy as about 25,000 households [9]). According to recent Gartner estimates[1], ICT industry accounts for approximately 2 percent of global carbon dioxide (CO2) emissions, a figure equivalent to aviation. Almost 90% of young Americans are "always connected". This also impacts energy consumption, e.g., in the Netherlands, total electricity consumption increased between 2006 and 2008 by 12%. Without focused environmental strategies, ICT related energy consumption is expected to grow by 73% until 2020 (instead of the targeted 26%).

IT systems and data centers are migrating towards a service-oriented approach in which the available computing resources are shared by several types of users and organizations. In such systems, the software is accessed as-a-service and computational capacity is offered on demand to customers who share a pool of IT resources in-the-cloud [5]. The software as-a-service (SaaS) model provides significant economies of scale, affecting the energy efficiency of data centers [1].

Environmental strategies are already a reality in non-IT domains. For instance, Sweden recently introduced on many supermarket products an indication of the CO2 emission related to their production process [8], to change alimentary habits toward more environmental friendly products, similar to what already happens for choosing cars and house-hold equipment. As another example in the more traditional hardware/embedded software domain, some companies already commercialize devices that are plugged in the electricity socket and allow users to monitor and gain detailed understanding of their energy consumption.

Adoption is much more difficult in intangible domains like software industry for at least two reasons. First, it is a challenge to increase *process awareness*, i.e. convince decision makers of the urgency of adopting ecological strategies in the processes of managing their IT portfolios and bring IT practitioners to adopt innovative ecological models in the way they engineer SBAs, i.e. in the development process. Second, we urgently need to change the way users exploit SaaS, by making them realize their energy consumption and suggest alternative consumption models. I.e. we need to create *people awareness*. Third, it is difficult to make explicit how SBAs should be engineered to become "greener", i.e. to define and implement how ecological strategies can be adopted by service-oriented software (*service awareness*). Awareness can be increased with a SaaS model, by offering "green metrics" to measure the level of greenness of software services, and incorporate in SBAs ready-to-use "feedback services", i.e. services that give feedback on the carbon footprint of SBAs and of their end-users.

In this paper we describe how service-oriented software can positively influence the software impact on the environment, i.e. we introduce the concept of *green problem areas*, defined as green issue categories that have to be resolved to achieve energy efficient software services and SBAs.

[1] Source: `www.gartner.com/it/page.jsp?id=503867` [15 September 2010].

As a result of literature scan and discussions with both researchers and practitioners in the field, we have distilled three green problem areas (further discussed in Section 2): the direct environmental impact of executing software services (service awareness); the indirect environmental impact of their development process (process awareness); and the use of services to monitor and measure the two above, so to create people awareness). Thanks to this holistic overview of green problem areas in SO we devised an approach, called Service Greenery, to address them in a structured way, as presented in Section 3 and based on previous work [4]. Conclusions and directions for future research conclude the paper.

2 Green Problem Areas in Service Oriented Software Engineering

Research efforts are needed to investigate the role of software services in making enterprises "greener". To this end, business strategies decided at the enterprise level must be aligned with green strategies (as illustrated in Fig. 1). Only in this way ecologic issues are incorporated in the strategic plan of the organization and transformed into business opportunities.

Green strategies can address three green problem areas, as illustrated in the Figure:

- **Create process awareness** by having a more sustainable (or agile) development process. Green strategies should influence (cf. Fig. 1) the process

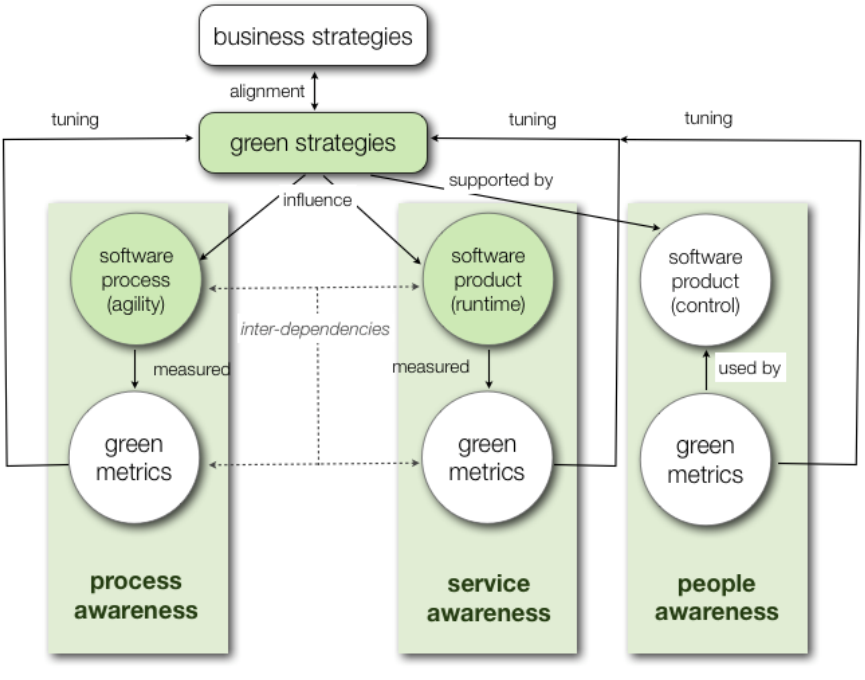


Fig. 1. Green Problem Areas in Service Orientation

followed to develop software services and SBAs (i.e. software process). By defining suitable green (process) metrics we can measure the current level of greenness of the processes, and tune the related green strategies to optimize it, making sure to achieve the target environmental objectives of the organization.

- **Create service awareness** by delivering energy-efficient software services and SBAs. Green strategies should influence (cf. Fig. 1) the direct impact of executing the software services/SBAs to render them energy-efficient and hence lower their own carbon footprint at runtime. To create service awareness green (service) metrics provide the necessary feedback to measure the direct environmental impact of software services and SBAs.
- **Create people awareness** by putting in place software especially conceived to control the impact of IT and its actors on environmental issues. Green strategies can be supported by (cf. Fig. 1) SBAs which are used to control some eco-friendly systems (e.g. SBAs controlling smart grids), to control the impact of individuals (e.g. SBAs regulating energy consumptions in smart homes), or to create awareness about some environmental issues (e.g. printing budgeting). In the first two cases we need to measure the process (respectively product runtime) impact on the environment (by use of especially conceived green metrics). To create people awareness, SBAs can use green metrics to realize the decided upon green strategies.

In summary, results of measurements can inject a tuning process to evolve the green strategies of an organization, and eventually determine the adaptation of the business strategies. The ability to define *sound* green metrics for process, service and people awareness is essential to realize this green vision. While some work is being done in this respect (e.g. [1]), most metrics address the environmental impact of IT rather than the software/services using it [7]. We regard the definition and experimentation of green metrics for SOSE in the three green problem areas as an important and mostly unexplored research area.

3 The Service Greenery Approach to Green SBAs

To increase both process, service and people awareness we propose an approach to create what we call *service greenery*, i.e. a portfolio of green software services accessible on the Web as-a-service. The service greenery supports the iterative, incremental approach to increasing sustainability, as illustrated in Fig. 2. This is centered around two types of services (central part in the Figure):

- **Environmental strategies** as-a-service, which set out the user's/organization's goals and actions for achieving a sustainable environment. Examples of generic organizational strategies are 'achieve tighter customer relationships' or 'become more standard'. Equivalent green strategies could be 'achieve carbon neutral data traffic' or 'decrease the company's global carbon footprint by x% before year y'. Such strategies will be defined in terms of the user's/organization's business processes. Accordingly, users/organizations can use environmental strategies as instrument for lowering carbon footprint, and at the same time keep them aligned with the supporting IT services/SBAs.

- **Green metrics** as-a-service, which measure the actual carbon footprint of SBAs. In this way, the user/organization can monitor real-time the SBA environmental impact and provide feedback to end-users and companies. This will increase people awareness (by leading users to opt for options consuming less energy) as well as service awareness (by quantifying the current carbon footprint).

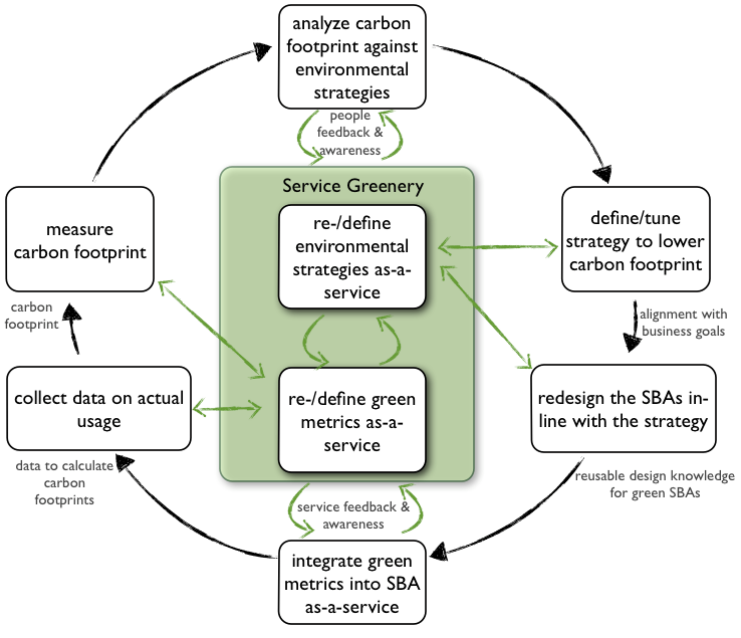


Fig. 2. Central role of the Service Greenery to redesign environmental-aware SBAs

- The **service greenery** will make available environmental strategies as-a-service and their supporting green metrics as-a-service. It will trigger use and reuse of the green metrics, create incentives to tailor strategies and measures to business goals, hence re-populating the service greenery with new resources. The ultimate goal is to make service oriented software "greener" by initiating the adoption of environmental strategies offered as-a-service, continuously measuring the achieved level of greenness, and tune the strategies to increase it.

4 Conclusions and Research Directions

In this paper we introduce three problem areas that frame the issue of creating environmental awareness in SOSE: the creation of process awareness, service awareness and people awareness. We also give a holistic overview of how enterprise business strategies and green strategies should be aligned. Green strategies influence process and service awareness, and the application of green metrics trigger the tuning of the

strategies. People awareness supports the achievement of green (and hence business) strategies thanks to the use of green metrics creating such awareness.

To address the green problem areas we also proposed an approach based on the SaaS model. The service greenery should offer a portfolio of services to measure relevant factors in the SOSE process and the level of greenness of software services/SBAs, as well as create people awareness.

No need to say that by making the ingredients of the service greenery available as reusable assets we can identify innovative ways to greenify the industry IT portfolio; we can increase process-, service- and people awareness thanks to continuous measurement & feedback; we can disseminate reusable green knowledge offered to researchers and practitioners.

While the three green problem areas could hold for any software, too, we study them specifically for service-oriented software, as more and more enterprises are migrating their software assets to SOA technologies. As future work we will study further the green factors belonging to each area, and their dependencies. This will help us in defining the relevant related green metrics.

Acknowledgments. This research has been partially funded by the European Community's Seventh Programme FP7/2007-2013 under grant agreement 215483 (S-Cube).

References

1. Ferreira, A.M., Kritikos, K., Pernici, B.: Energy-Aware Design of Service-Based Applications. In: Baresi, L., Chi, C.-H., Suzuki, J. (eds.) ICSOC-ServiceWave 2009. LNCS, vol. 5900, pp. 99–114. Springer, Heidelberg (2009)
2. Gu, Q., Lago, P.: Exploring Service-Oriented System Engineering Challenges: A Systematic Literature Review. Journal of Service Oriented Computing and Applications 3, 171–188 (2009)
3. Lago, P.: Establishing and Managing Knowledge Sharing Networks. In: Ali Babar, M., Dingsoyr, T., Lago, P., van Vliet, H. (eds.) Software Architecture Knowledge Management - Theory and Practice, pp. 113–131. Springer, Heidelberg (2009)
4. Lago, P., Jansen, T., Jansen, M.: The Service Greenery - Integrating Sustainability In Service Oriented Software. In: International Workshop on Software Research and Climate Change (WSRCC), co-located with ICSE, 2 pages (2010)
5. Dikaiakos, M.D., Katsaros, D., Mehra, P., Pallis, G., Vakali, A.: Cloud Computing: Distributed Internet Computing for IT and Scientific Research (Guest Editors' Introduction). IEEE Internet Computing 13, 10–13 (2009)
6. Papazoglou, M.P.: Web Services: Principles and Technology. Prentice-Hall, Englewood Cliffs (2007)
7. Gude, S., Lago, P.: A Survey of Green IT – Metrics to Express Greenness in the IT Industry, Technical report VU University Amsterdam (August 2010)
8. Rosenthal, E.: To Cut Global Warming, Swedes Study Their Plates. New York Times (October 22, 2009)
9. Kaplan, J.M., Forrest, W., Kindler, N.: Revolutionizing Data Center Energy Efficiency McKinsey & Company (August 2008)

Towards Green Business Process Reengineering

Alexander Nowak, Frank Leymann, and Ralph Mietzner

University of Stuttgart, Institute of Architecture of Application Systems,
Universitaetsstrasse 38, 70569 Stuttgart, Germany
firstname.lastname@iaas.uni-stuttgart.de

Abstract. Information and communication technology has experienced a vast development and increased usage over the past few years. This development again yields to increasing energy consumption. In this paper we provide a research agenda that picks up this serious development and suggests first approaches how holistic energy efficiency could be introduced in enterprises without neglecting a company's performance and competitiveness. We propose *green Business Process Reengineering* as one opportunity to make further development more sustainable with respect to the resources of our environment.

Keywords: Green BPR, Cloud Computing, Energy Efficiency, Green IT.

1 Introduction

Energy-efficiency in information and communication technology (ICT) has become an important issue over the last years. The highly increasing usage of ICT calls for intelligent energy-efficient technologies to handle and decrease the worldwide energy consumption. According to Amazon's estimations [1] the expenses for energy-related costs amounts to 42% of the total amount spend for cost and operating of servers (based on a three year amortization schedule). Even more critical is the fact that current data centers spend about 60% to 70% of their total expenses just for cooling their ICT equipment [2]. As more and more business processes in enterprises are supported by corresponding IT, energy efficiency of the IT infrastructure and whole business processes becomes an important target to cut costs and to improve the green image of an enterprise. Due to this dramatically increasing consumption of energy a few approaches have emerged targeting the optimization of energy consumption of hardware as well as data centers as a whole. Different vendors introduced techniques to throttle down their CPUs or turn off hardware parts that are idling, for instance [3]. The collection of all techniques, methods, and technologies aiming at an energy-efficient resource usage within modern ICT is summarized under the term *Green IT*.

The main problem within this development is that the currently proposed technologies are only designed for their particular and isolated scope but, however, do not provide a holistic view on a companies' complete ICT. As companies mostly consist of inter-operating processes formed as networks of inter-organizational partners, changes in a single processes may have impact on different parts of the network and thus a couple of other processes. Therefore, a holistic perspective

E.M. Maximilien et al. (Eds): ICSOC 2010 Workshops, LNCS 6568, pp. 187–192, 2011.

focusing on all aspects of reducing energy consumption, i.e., people, processes, and infrastructures is needed. In this paper we propose to use the Business Process Reengineering (BPR) methodology to tackle the gap of missing interconnection between existing stand-alone solutions and the holistic reduction of energy consumption in modern ICT. We call this approach *green Business Process Reengineering* (gBPR). In order to show how gBPR can be realized within enterprises in a holistic way, we provide a first taxonomy containing information on which parameters to measure the 'greenness' of a business process. We then show how gBPR can be realized and how it influences an organizations' business processes, IT infrastructure, and organizational structure.

The remainder of this paper is structured as follows: Section 2 summarizes the state of the art concerning Green IT and BPR and identifies the missing gaps. Section 3 proposes the vision of combining Green IT and BPR and discusses the chances and risks that may occur as well as some areas that need further research. Section 4 summarizes the proposed vision and proposes additional topics for future research.

2 Background and Related Work

Green IT in general focuses on a growing sustainability in modern ICT [4]. It characterizes the efficient use of environmental and computing resources with the primary objective to account for the 'triple bottom line': *people*, *planet*, and *profit* [5]. Considering the nature of modern ICT systems that rely on people, networks and hardware, the elements of Green IT may focus on a broad range of items such as customer satisfaction, management and organizational restructuring, regulatory compliance, virtualization of servers, energy use, or thin clients [6]. This comprehensive understanding of Green IT is again one of the most important issues for future development. Therefore, in order to achieve an energy-efficient and sustainable growth in future ICT it is necessary to consider the following aspects: (1) organizational structures and customer demands (*people*), (2) reducing green house gas emissions in processes, software, hardware, and infrastructure (*planet*), and (3) ensure the competitiveness of an enterprise (*profit*). This needs to be incorporated in a combined and holistic manner. However, there is no such holistic approach available so far. There already exist several approaches to reduce power consumption in hardware [7], networks [8], and data centers [9]. Green IT, however, consists of more than optimizing the energy consumption based on technical progress. From a companies' point of view it is also important to involve the people that form a companies' organization, people that are obliged to use ICT systems and the customers that need to be satisfied. However, while optimizing the energy-efficiency it is also important to keep a company competitive.

First approaches towards Green-IT mostly concentrate on a single specific element that depends on the objective that has to be optimized. They can fundamentally be distinguished between infrastructure and process optimization. Liu et al. [10], for instance, take the energy consumption of physical servers to derive a cost function for the optimal positioning of virtual machines within data centers. Hence, they directly concatenate energy-efficiency with costs which mostly drives companies' efforts in Green IT. Berl et al. [3] aim at the reduction of green house gases with respect to

energy-efficient infrastructures. For their proposed energy-efficient cloud computing infrastructure this aspect is also reduced to the power consumption in computing, storage and communication. A wider perspective is proposed by Ghose et al. [11] through annotating tasks of business processes with their carbon dioxide equivalent (CO2-e). This enables the optimization of the process structure regarding CO_2 emissions. Thus, the approach by Ghose et al. [11] is a first step to apply business process reengineering (BPR) methods to optimize business processes regarding their CO_2 emissions. This is in line with traditional BPR that has been long used to improve the efficiency of a companies' core business processes by fundamentally and radically redesigning them [12]. These radical changes enable companies to enhance their productivity and competitiveness by adapting state-of-the-art organizational structures and business processes that satisfy customer demands [13]. When thinking about Green IT we are faced with a very similar situation. However, as we showed in this chapter, current approaches towards Green IT do not combine BPR methods with infrastructure reorganization. Consequently, the vision we outline in the following aims at combining these two aspects.

3 A Framework for Green Business Process Reengineering

The previous sections highlighted the need for developing a comprehensive approach to support an energy-efficient ICT that involves the whole ecosystem of a company. Business Process Reengineering is based on such a holistic improvement approach and therefore provides a good starting point for considering energy consumption within and across complete enterprises. To make the effects of 'greenifying' modern ICT visible and measurable, it is important to define criteria that allow measure the differences between both approaches. We refer to such criteria as *Key Ecological Indicators* (KEI) as they are in fact special Key Performance Indicators (KPIs). Due to the holistic point of view KEIs may comprise various elements such as energy consumption, regulatory compliance, carbon footprint, the location of hardware, water consumption, sustainability, or recycling, for instance. It is important not to reduce this KEI only to a process level but to consider them across entire ICT systems. To illustrate our proposed framework we use a running example describing the company *Deal inc.* that buys goods from various manufactures and sells them to customers worldwide. Today, *Deal inc.* manages their shipment services (i.e., printing, packaging, shipping etc.) using an application running in their own datacenter.

3.1 Optimizing Business Processes

With respect to Green IT, the process optimization definition of BPR needs to be extended as changes in business process are faced to influence both KEI and KPI depending on the companies' attitude towards Green IT. The main challenge for novel approaches is to obtain processes that are functionally equivalent or at most similar to the former ones but maximize KEIs (i.e., are 'greener') when executing them. Therefore, new process models need to be derived by optimization algorithms. These Optimization algorithms must take into account not to worsen existing KPIs on process level (or only worsen them to a degree that is acceptable by the enterprise).

Several process optimization techniques known from BPR are possible including: (i) Dynamic binding of services implementing a process activity, based on their KEIs (ii) optimization of control flow and data-flow (i.e. introduce or remove parallelization) to optimize the KEIs of the whole process (iii) addition, removal or modification of (groups of) activities. In future work we will extend the work of [11] to identify suitable optimization techniques. In our running example, *Deal inc.* may decide to outsource their postal service by using ePost that is provided by Deutsche Post AG. This supports *Deal inc.* to digitally send their mail to ePost service which routes the mail electronically near the recipients' location, preventing the pick-up of the mail by a truck and thus decreasing emissions. This is an example of optimization technique (i) and (iii) as different services are bound and additional activities must be added or existing ones must be changed.

3.2 Optimizing Process Infrastructures

Another possibility to reduce the energy consumption of ICT systems is to optimize the underlying infrastructure. Introducing Cloud computing techniques, for instance, is one commonly referenced optimization that enables enterprises to decrease their carbon footprint by providing increased server utilization and provisioning of resources on demand [10,14]. This results in energy savings on both, customers and providers side. While improving server utilization increases the efficiency of data centers, cloud infrastructures also enable various opportunities to save energy on consumers' side. Think, for instance, about management tools that smartly coordinate the selective operation of computing resources. Constrained on given Quality of Services (QoS) such tools may queue upcoming requests and only deploy a particular computing resource at a specific time period of each day or when a defined threshold of requests is reached. Note, that this also requires a close association to the underlying business processes which purport their individual QoS. Due to the switching of computational power from local sites into cloud infrastructures thin clients also provide suitable opportunities in reducing energy consumption.

Let us recall the *Deal inc.* example. The CIO of *Deal inc.* may decide to switch computing resources from their own datacenter to a cloud provider. This allows the company to deploy exactly the amount of resources they actually need. They no longer need to provide a high amount of computing resources that most of the time sit idle to cover peaks of order requests, for instance.

3.3 Green Business Process Reengineering

We have shown that existing technologies are able to support Green IT and provide already today a wide range of opportunities for energy-efficient computing. However, through focusing only on particular areas of concern, the full potential for the optimization of KEIs is not used. Companies need to consider energy efficiency like any other aspects they use to improve their business processes on a holistic base. We therefore propose to use gBPR as a methodology to comprise the stand-alone approaches for fundamentally and comprehensively redesign a company's processes, infrastructure and organization towards an energy-efficient ecosystem. This includes redesigning processes in a way so that they can make use of optimized infrastructure,

such as Clouds. As a result, the new approach gBPR comprises means to enable enterprises to (1) sustainable redesign their business processes based on global KEIs, (2) integrate BPR and infrastructure optimization techniques (3) consider the trade-off between KEIs and KPIs, (4) avoid organizational inefficiencies through integrating the 'human factor', and (5) optimize the energy consumption beyond the companies' boundary by taking externally sourced services into account.

Taking our example, *Deal inc.* now wants to use gBPR to lower their carbon footprint even more. First, they decide to restructure their shipping process. The shipper picks up the charge only once instead of three times a day. This will reduce, e.g., the KEI "CO_2 *emission*" of the shipping about two-thirds but also ensures next day delivery provided by the shipper (KPI). Secondly, they decrease the communication with the shipper to a once per day basis as the charge is picked up only once a day. Through the new Cloud infrastructure of *Deal inc.*, the company is now able to start up the necessary computing resources to process the shipping paperwork once a day and shut them down after the communication with the shipping company is completed. Through the combination of process restructuring and using a more elastic infrastructure, *Deal inc.* can now further optimize their overall KEIs.

3.4 Chances and Risks

The need for energy-aware computing is beyond all questions and our proposed approach has shown how current technologies can be utilized to realize such energy-aware environments. However, when performing gBPR there are also several risks that need to be taken into account. First, we need to consider that due to the globalization of economies, companies are always forced to act in a profitable way concerning specific quality of services demanded by customers. Therefore, we need a new systematic approach handling the trade-off between a company's traditional KPIs such as 'throughput', 'availability' or 'costs' and the objectives of Green IT. Second, switching proprietary IT into cloud infrastructures influences the organizational structure of a company. Similar to traditional BPR the 'human factor' needs to be integrated within the migration process [15] to avoid organizational inefficiencies [16]. This also includes the serious aspect of losing business through moving to the cloud. Third, the increasing amount of data volume occurred from new infrastructures needs to be tackled. Novel approaches are needed to transfer data in more intelligent ways by providing the advantages from a more decentralized computation paradigm, for instance.

4 Conclusion

We have provided an overview that illustrates various aspects for future research in the emerging area of energy-awareness in modern enterprises. We pointed out the crucial demand for a holistic optimization viewpoint and how individual existing methods and technologies are important parts in reducing carbon emission. To strengthen the coherence between the various existing individual approaches we proposed a methodology that supports for a holistic energy-aware approach within and across enterprises, named *green Business Process Reengineering*. Future work

will capture the development of first patterns on how green business process reengineering is able to be applied to enterprises.

References

1. Hamilton, J.: Cooperative Expendable Micro-Slice Servers (CEMS): Low Cost, low Power Servers for Internet-Scale Services. In: 4th Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, pp. 1–8 (2009)
2. Malone, C., Belady, C.: Metrics to Characterize Data Centre & IT Equipment Energy Use. In: Digital Power Forum, Richardson, TX, USA (2006)
3. Berl, A., Gelenbe, E., Girolamo, M., Giuliani, G., Meer, H., Dang, M., Pentikousis, K.: Energy-Efficient Cloud Computing. Computer Journal 53(7) (2010)
4. Marinos, A., Briscoe, G.: Cloud Computing. LNCS, vol. 5931. Springer, Heidelberg (2009)
5. Elkington, J.: Cannibals with forks: the triple bottom line of 21st century business. New Society Publishers (1998)
6. Williams, J., Curtis, L.: Green: the new computing coat of arms? IT Professional 10(1), 12–16 (2008)
7. Intel whitepaper 30057701: Wireless Intel SpeedStep Power Manager: optimizing power consumption for the Intel PXA27x processor family (2004)
8. Gelenbe, E., Silvestri, S.: Reducing Power Consumption in Wired Networks. In: 24th Annual International Symposium on Computer and Information Sciences, London, pp. 292–297 (2009)
9. Beloglazov, A., Buyya, R.: Energy Efficient Resource Management in Virtualized Cloud Data Centers. In: 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, Melbourne, pp. 826–831 (2010)
10. Liu, L., Wang, H., Liu, X., Jin, X., He, W., Wang, Q., Chen, Y.: GreenCloud: A New Architecture for Green Data Center. In: 6th international Conference on Autonomic Computing and Communications (industry session), New York (2009)
11. Ghose, A., Hoesch-Klohe, K., Hinsche, L., Le, L.: Green Business Process Management: A Research Agenda. Australasian Journal of Information Systems 16(2), 103–117 (2009)
12. Hammer, M., Champy, J.: Reengineering the Corporation: A Manifesto for Business Revolution. Harper Business, New York (1993)
13. Guha, S., Kettinger, W., Teng, J.: Business Process Reengineering. Journal of Information Systems Management 10(3), 13–22 (1993)
14. Armbrust, M., Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., Zaharia, M.: Above the Clouds: A Berkeley View on Cloud Computing. Technical Report (2009)
15. Yanosky, R.: From Users to Choosers: The Cloud and the Changing Shape of Enterprise Authority. In: Katz, R. (ed.) The Tower and the Cloud, Educause, pp. 126–136 (2008)
16. Khajeh-Hosseini, A., Greenwood, D., Sommerville, I.: Cloud Migration: A Case Study of Migrating an Enterprise IT System to IaaS (2010), http://arxiv.org/abs/1002.3492

Business Process Improvement in Abnoba

Konstantin Hoesch-Klohe and Aditya Ghose

Decision Systems Lab (DSL),
School of Computer Science and Software Engineering,
University of Wollongong

Abstract. A key element of any approach to meeting the climate change challenge is the ability to improve *operational efficiency* in a pervasive fashion. The notion of a *business process* is a particularly useful unit of analysis in this context. In the *Abnoba framework*, we enable business process management (and in particular, business process design/re-design) with explicit support for the environmental sustainability aspects of processes. This article extends our earlier work on the framework by introducing and elaborating a machinery for (semi-)automated process re-design discovery. The machinery leverages a library of process snippets/fragments, used to replace fragments from the library with fragments of the process design, such that the process re-designs meets *functional-*, *process provisioning-*, and *compliance requirements* and the sustainability profile is improved.

Keywords: automated process improvement, process provisioning, semantic annotation, green/sustainable BPM, Abnoba.

1 Introduction

A key element of any approach to meeting the climate change challenge is the ability to improve *operational efficiency* in a pervasive fashion. The notion of a *business process* is a particularly useful unit of analysis in this context. In the *Abnoba framework*, we enable business process management (and in particular, business process design/re-design) with explicit support for the environmental sustainability aspects of processes. This article extends our earlier work on the framework [1,2,3] by elaborating a machinery for (semi-)automated process improvement, using a library of semantic annotated process snippets. Modeling organizational behavior in a business process we are interested in finding process design alternatives, which constitute an improvement, with respect to certain measures (e.g. CO2 emission), over the as-is design. Various techniques have been described in the literature for process improvement, see Reijers and Mansar [4] for a survey. While existing work primarily provides process re-design guidelines to the analyst, less attention is given to automated process re-design/improvement machineries. Existing machineries for automated process improvement (e.g. [5]) mainly focus on parallelizing activities. While parallelizing activities can have a positive impact on the process cycle time, the resulting process re-design is not necessarily superior with respect to other criteria like the sustainability profile.

E.M. Maximilien et al. (Eds): ICSOC 2010 Workshops, LNCS 6568, pp. 193–202, 2011.

Furthermore, the number of applicable process re-designs is limited by the size of the search space constituting all possible process re-designs, which can be devised by parallelizing activities of the process design under consideration. By extending the size of the search space, we can increase our chances to find more preferable process re-designs. We extend the search space by devising a library of process snippets, which constitute best practice process designs and fragments for a given domain. Essentially, we seek to replace process fragments of an as-is process design with other fragments (drawn from the library) in a manner that ensures that the *original goals of the process are still realized* (the same desired functional outcome is achieved), but the sustainability profile of the resulting process design is improved. Such a machinery requires the process designs (and fragments in the library) to be annotated with semantic effects and a machinery for effect accumulation. For this, we build upon of the ProcessSEER framework [6,7,8]. However, when introducing new process fragments (from the library) to the as-is process design we must also ensure that the process re-design, can still be executed in the organizational *resource context*. In this context, we describe a rich resource model, a machinery for correlating resources with process designs and finally the resulting concept of (abstract and concrete) process provisioning. These results extend and improve upon earlier work on resource modeling [9,10,11,12,13]. In addition, it must be ensured that the process re-design does not violate any *compliance requirements*. Business process compliance for semantic annotated process designs can be found in [7,6], we omit details for brevity. In the remaining article we first (Section 2) describe work done on resource modeling and introduce the concept of process provisioning, before (in Section 4) we elaborate the proposed machinery for process improvement and finally conclude (in Section 5) the article.

2 Resource Modelling

In this following we extend our previous work on resource modeling [2], by distinguishing between different resource and relationship types. In addition to a "Use" relationship (denoting that one resource is used by another), we allow relationships of type "IS-A" and "Whole-Part", which enables us talk about resources on different level of abstraction. Furthermore, a resource can be of type "schedulable resource class", "unschedulable resource class", and "resource instance. Both the new resource- and relationship types are taken from Podorozhny et al.[9], who introduces and applies a resource management system in agent and activity coordination. The new relationship and resource types are described in more detail as follows:

– **Resource Types:** A *resource instance* is a unique representation of an entity from the physical environment of discourse. It therefore denotes a specific resource in the organization. A *schedulable resource class* is a set of resources, which are substitutable for each other and can be allocated to an activity. An *unschedulable resource class* is used to structure resources with

similar properties that are not schedulable. We use this class to describe resources like electricity, which cannot directly be used by an activity, but are rather indirectly via electrical devises.
- **Relationship Types:** A *IS-A relation* is used to denote that entities can share a set of attributes, where the attributes of one entity are inherited by all specializations for that entity. The *Whole-part relation* is another abstraction mechanism, where the linked resources constitute part of an aggregated resource. The *Use relation* (Podorozhny et al. uses the term "requires relation") denotes that a resource entity requires another entity to be used.

We formally define this model as follows: A resource model RM is denoted by a labeled digraph $\langle V, E, \Omega_V, \Omega_p \Omega_E, f_V, f_E, f_p \rangle$

- V is an partially ordered set of vertices $v \in V$.
- E is a set of edges such that $(x, y) \in E \subseteq V \times V$, where $x, y \in V$ and x is pointing to y.
- Ω_p is a set of properties, denoted by a triple $\langle id, type, value \rangle$, such that id is the unique identification of a property, $type$ denotes the type of property, and $value$ is a set of values.
- Ω_V is a set of vertex labels where each label is a tuple $\langle id, type \rangle$, such that id is the unique identification of the label and $type$ denotes the type of resource where $type \in$unschedulable resource class, schedulable resource class, resource instance;
- $f_p : \Omega_V \rightarrow 2^{\Omega_p}$ is a function mapping labels to elements in Ω_p.
- Ω_E is a set of edge labels denoted by $\langle id, type \rangle$, such that id is the unique identification of an edge label; $type$ is the type of the edge where $type \in$ {IS-A, Whole-Part, Use};
- $f_V : V \rightarrow \Omega_V$ is a function mapping vertices to a vertex label.
- $f_E : E \rightarrow \Omega_E$ is a function mapping edges to an edge label.

In addition the following rules over the model must hold. (1) Subgraphs of RM, where all edges are of type "IS-A" must be a tree. (2) There exist no two edges of type "IS-A" and "Whole-Part" between the same two nodes.

3 Process Provisioning

We refer to the correlation of resource models and process designs as the provision of the process design by the resource model. We consider *process provisioning* in two different ways. In *abstract process provisioning* we correlate an activity in the process design with a resource in the resource model. The resource in question might be arbitrarily abstract, e.g. a "printer" resource, or a more specific "inkjet printer" resource, or an even more specific "inkjet printer model x of manufacturer y" resource. By establishing an *abstract provisioning relation* between a task in a process design (say "print report") and any of these resources we merely assert a provisioning relation (e.g. the "print report" activity shall use the "inkjet printer" resource, or an even more specific resource). In *concrete*

process provisioning, we annotate the abstract provisioning relation with specific quality of service (QoS) requirements. For instance, we might establish a *concrete provisioning relation* between the "print report" activity and an "inkjet printer" resource which admits a configurable image resolution setting by stating the minimum image resolution requirements.

Technically, we correlate process designs with resource models by annotating each activity of a process design with an *expression*. The annotated expression references *one or more* resource entities, where each reference is accompanied with a (potentially empty) set of QoS requirements. Each QoS requirement is described by a triple $\langle QoS, value, 1\rangle$, where QoS denotes the QoS measure of consideration, *value* the required value, and 1 the most preferred value (as in the algebraic c-semiring structure[14], which we use in [3] to handle multidimensional qualitative and quantitative measures). The existence of a resource model permits us to explore a range of provisioning options. An *activity can be provisioned* by by a set of resources R if (1) it requires R' resources and all elements of R' are also an element of R or of an specialization in R. (2) R' meets all QoS requirements stated in the provisioning relation. A resource meets a QoS requirement if the QoS value is equal or between the required value and the most preferred value (1). A *process design can be provisioned* if all its activities can be provisioned.

It can be observed that the provisioning relations serve as constraints, restricting the space of applicable process designs and applicable resource models. This is particular important in the face of process re-design and optimization, where it must be ensured that the *process re-design can still be provisioned* by the resource context. Similar, for any changes on the resource context (e.g. exchanging resources, or deploying a process design in a new resource environment) provisioning must be ensured. In the following we will focus on process design improvement, omitting resource optimization, which we seek to address in future work.

4 Business Process Improvement

Process improvement must involve process re-design to obtain processes that achieve the same (functional) goals, while minimizing the environmental impact (and potentially other non-functional criteria as well). To achieve this, we need (1) the ability to annotate process designs with detailed specifications of functional effects (2) the ability to correlated organizational goals with process designs (3) the ability to search for process re-designs through a space of alternative process designs.

4.1 Preliminaries

This subsection provides the preliminaries on annotating and accumulating functional effects, correlating goal models with process models, and change patterns for design time change on process designs.

Annotating and Accumulating Functional-Effects. A process design can be semantically enriched by specifying the outcome of each activity. This is done by annotating each activity with its immediate effects (post conditions), denoting the consequence of executing the respective activity in some state in the environment. Immediate effects are represented as logic statements in conjunctive normal form. We pairwise accumulate effects to receive a cumulative effect. Pairwise accumulation is done by set union the effects of two sequentially linked activities, such that the combined set of effects is consistent (we can view sentences in conjunctive normal form as sets of clauses, without loss of generality). If the resulting set is not consistent the cumulative effects consists of the effects of the second activity and as much effects of the first activity as can be consistently included (effects are overwritten/undone by the second activity). Furthermore, all cumulative effects must be consistent with a background knowledge base (KB) containing rules (e.g. for compliance). Note that this KB can also be used to state pre-conditions (conditions that must hold before the activity can be executed) in a centralized manner. Effect accumulation in parallel environments is done by pairwise accumulating the cumulative effects of each branch with the effects of the activity directly following the AND-join and combining these sets via set union. A detailed description of this machinery can be found in [6,7,8]. The functional outcomes of a process are its cumulative effects at the end-event. A process design can have multiple cumulative effects outcomes one for each distinct execution path.

Correlating Goals with Process Models. A goal is an objective the organization seeks to achieve. It can be formulated at different levels of abstraction ranging from abstract goals (e.g. strategies) to further decomposed goals. A rich body of knowledge on goal modeling can be found in the goal-oriented requirements engineering literature. We list besides others [15,16]. Correlating process models with goal models is crucial to ensure that the process environment, being under change, maintains to satisfy the organizational objectives. Koliadis and Ghose[17] proposes the GoalBPM methodology for relating business process models (in BPMN) to functional goals (in KAOS[16]) via satisfiability links. These links are drawn between the process model and goals of the goal model, denoting the goals satisfied by the process design. A goal is represented as logic statement and is in its simplest form a single literal α, or conjunction of literals $\{\alpha \wedge \beta\}$, for example "package send" and "bill send".

A process design *satisfies* the organizational objectives if it satisfies *all* correlated goals. A correlated goal is satisfied by a process design if all its cumulative effects at the end-event entail the goal. For example the cumulative effect $\{\alpha \wedge \beta \wedge \gamma\}$ entails (denoted by $\models$) the goal $\{\alpha \wedge \beta\}$). The implication of requiring all cumulative effects to entail a goal is that each path through the process design must lead to goal satisfaction. It is obvious that a path, which does not satisfy the objectives of a process is superfluous. Note that a correlated goal does not have to denote "positive" outcomes only. For example, a process "credit card application" might have the goal "application processed" and the background

KB tells us that either the cumulative effect "credit card issued" or 'credit card declined" satisfies the goal.

Process Change. Weber et al. [18] introduce high-level change patterns and change support features to assess existing process aware information systems in their ability to deal with process change. The high-level change patterns are divided into patterns supporting structural process adaption (adaption patterns) and patterns for built-in flexibility at predefined regions (e.g. to add information during run-time). We focus on the adaption patterns, since our focus is on design time optimization. Equally to Weber et al., we use the term *process fragment* to denote atomic activities, hammocks (sub-process graphs with a single entry and exit point), and complex activities (sub-process). The change patterns *delete process fragment* from process design and *replace process fragment* by another fragment, are of interest to us. Applying these change patterns on a process design we are able to discover potential process re-designs. Due to the nature of process fragment (single entry and exit point) the well-formedness of the process re-design is ensured.

4.2 A Machinery for Process Improvement

In this subsection we build on the preliminaries of the previous subsection and introduce a machinery for discovering process re-designs using a library of process fragments. Figure 1 provides an overview of the procedure. The original process design (upper left corner), is (1) disassemble into its process fragments (lower left corner). (2) The change pattern "delete process fragment" is applied to delete potentially obsolete fragments. (3) search for substitutable fragments in the library (lower right corner). (4) Replace substitutable fragments and check whether the resulting process re-design achieves the same desired functional outcome. (5) Order process re-designs according to their environmental profile. (6) Repeat previous steps until termination criteria is met.

Disassemble Process Design. In the first step we disassemble the as-is process design in a straight forward fashion into its process fragments (details omitted for brevity).

Delete Obsolete Fragments. In the second step we identify and delete obsolete fragments, by checking for each identified process fragment, whether it can be deleted, such that the resulting process design still satisfies the correlated goals (desired functionality). We do this by accumulating the semantic effects to get the cumulative effects. We then check whether the cumulative effects entail all correlated goals (as described in subsection 4.1). If the cumulative effects entail the correlated goals the process design is kept and the previous step repeated until no more fragment can be deleted.

Find Substitutable Fragments. In the third step we seek to find a substitutable process fragments, for each remaining process fragment, in a library

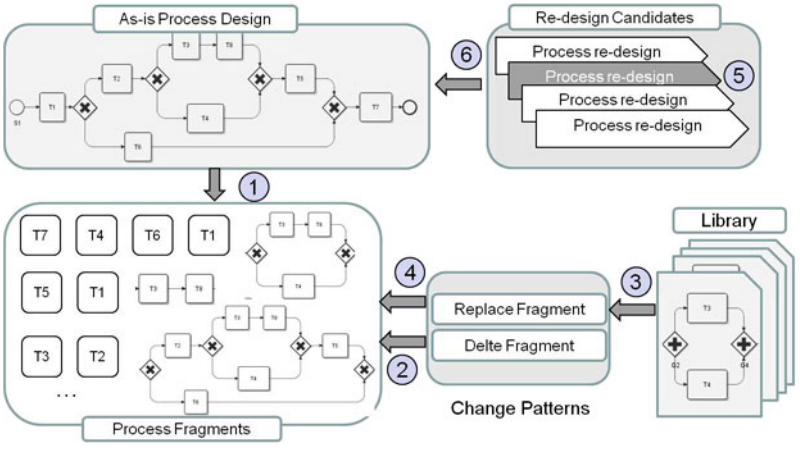


Fig. 1. Process Improvement

of process fragments. The library constitutes as a set of (semantic- and provisioning constraint annotated) process design snippets, denoting "best practice" solutions. The library is not restricted to process snippets, but could also include services (which can be seen as complex activities), derived from a service broker. Searching for replacements in the library can hence be utilized as an instrument for business functionality outsourcing. A process fragment p' is potential substitute for another process fragment p if and only if every terminal effect scenario of p is entailed by some terminal effect scenario of p'. For example, consider a process fragment with two sequentially linked activities T_n and T_m with a cumulative effect of $\alpha \wedge \beta$ at T_m. This process fragment is substitutable by an process fragment in the library with one activity T_n and the effect $\alpha \wedge \gamma \wedge \beta$, since $\{\alpha \wedge \gamma \wedge \beta\} \models \{\alpha \wedge \beta\}$.

Replace Substitutable Fragments. In the fourth step we replace each process fragment of the as-is design with a substitute in the library (if there exists one). Each such replacement denotes a potential process re-design, such that the number of identified substitutable fragments denotes the number of potential process re-designs. Due to the nature of process fragments the well-formedness of each such process re-design can be guaranteed. However, it is not guaranteed that the potential process re-design candidate has the intended functional (cumulative effect) outcomes and whether it violates any compliance requirements (details omitted for brevity). This is due to the fact that a fragment from the library can unintentionally add additional effects to the process design (e.g. γ in the example given above), which might conflict with effects of other activities, resulting in unintended cumulative effects and finally in non-satisfaction of correlated goals.

A process fragment p' is defined to be a *viable process re-design* for another process fragment p of a process design P if and only if: (1) For every cumulative

effect outcome s' in process design P' obtained by replacing p with p' in P, $s' \models G_p$, where G_p is the goal of process P. (2) P' does not violate any compliance requirements. (3) P' can be provisioned. We use the ProcessSEER machinery, described in subsection 4.1 to get the cumulative effects. Finally, all *potential* process re-designs that satisfy the correlated goals, can be provisioned, and do not violate any compliance requirements, are added to a *set of process re-designs.*

Please note that the effect accumulation procedure for parallel environments assumes that the cumulative effects of each branch are consistent - there exists no execution orders of activities in an parallel environment that leads to inconsistency. As pointed out by [8] such a scenario only occurs if the process is designed erroneous. Modeling activities in parallel we implicitly accept that the execution order between activities on different branches does not have an impact on the functional result of the process. However, in our case we cannot guarantee that the assumption still holds. We therefore have to determine all interleaving of a parallel environment to identify execution orders leading to undesired cumulative effects. This can be a computational expensive task for large process models. However, we can reduce the computational costs for most cases by devising a pre-checking procedure that catches *potential conflicts.* This is done as follows: Let us devise a set Ω for each branch $n, ..., m$ in a parallel environment, where Ω_n is the set of all effect elements of a branch n. A *potential conflict* is identified if $\Omega_n \cup ... \cup \Omega_m \cup KB \models \perp$. We only have to determine the cumulative effect of all possible interleaving of the parallel environment, if there exists a potential conflict. Although, we cannot always exclude this scenario, we note that we are in the game of design time improvement, where time constraints are less strict.

Order Process Re-designs. In the fifth step we order the process re-designs according to their sustainability profile, such that the most preferred process design can be identified. In [3] we leverage the algebraic c-semiring structure[14] to compare "green" QoS values of both *qualitative and quantitative nature.* Essentially, a comparison operator "$\oplus$" is defined such that $a \oplus b = a$ if a is more preferred as b ($a \geq b$, where $\geq$ is a partial order over all values of a measure). The process is straight forward for a *single measure* and a process re-design with a *single execution* path. For process designs with *multiple execution paths* we end up with multiple values for a single measure, one for each path. This is handled as follows. (1) For a QoS measure of *quantitative nature* we use the average value over all paths. This approach essentially place equal weighting on all paths. However, this approach might be erroneous due to the fact that some rarely executed scenarios might skew the mean value. To avoid this issue, we can consider the probability (if available) of taking a path through the process design during execution. Such figures can be derived from process logs or user experience.(2) For a QoS measure of *qualitative nature* we either chose the "best" or "worst" value as corresponding value. In cases where we want to compare process re-designs with respect to *multiple QoS measures*, we compose each QoS measure (denoted by a single c-semiring), where the composite structure is a c-semiring as well (proofed in [14]). Hence we can use the "$\oplus$" comparison

operator of the composite structure to compare the values. For a more detailed discussion see [14] and [3].

Repeat Previous Steps. In the sixth step the procedure is *repeated for all identified process re-designs* until a termination criteria is met (e.g. number of iterations, duration, no improvement after x amounts of iterations in a row). Please note that the described procedure is not complete in the sense that we cannot guarantee that the best possible process re-design has been identified. Nonetheless, we believe that the procedure can already provide significant improvements after a few iterations. Furthermore, the procedure can be interrupted at any time providing the user with the current order over the already identified process re-design suggestion.

5 Conclusion

In this article we elaborated a machinery for process improvement and the concept of process provisioning, being part of the *Abnoba framework* for green business process management. The process improvement machinery leverages a library of (best practice) process snippets to discover potential process redesigns, while ensuring that functional, compliance, and process provisioning requirements are met. This machinery will allow us to provide tool support for environmental aware semi-automatic process improvement. The described machinery behaves *semi*-automatic, because it does not consider interconnections (e.g. message flows) across and between pools and hence requires the analyst to correctly place these links. Nevertheless, we believe that the workload for the analyst can be decreased, and that further (potentially overlooked) process re-design alternatives can be revealed.

Future work is concerned with developing a thorough implementation for an industry evaluation. Furthermore, we seek to explore machineries for resource optimization and explore the interactions with the process improvement machinery.

References

1. Ghose, A., Hoesch-Klohe, K., Hinsche, L., Le, L.S.: Green business process management: A research agenda. Australian Journal of Information Systems 16(2) (2009)
2. Hoesch-Klohe, K., Ghose, A.: Towards Green Business Process Management. In: Proceedings of the 7th International Conference on Services Computing (Industry and Application Track) (2010) (to appear)
3. Hoesch-Klohe, K., Ghose, A.: Carbon-Aware Business Process Design in Abnoba. In: Maglio, P.P., Weske, M., Yang, J., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6470, pp. 551–556. Springer, Heidelberg (2010)
4. Reijers, H., Liman Mansar, S.: Best practices in business process redesign: an overview and qualitative evaluation of successful redesign heuristics. Omega 33(4), 283–306 (2005)

5. Netjes, M., Reijers, H., Aalst, W.: On the Formal Generation of Process Redesigns. In: Ardagna, D., Mecella, M., Yang, J. (eds.) BPM 2008 Workshop. LNBIP, vol. 17, pp. 224–235. Springer, Heidelberg (2009)
6. Ghose, A., Koliadis, G.: Auditing business process compliance. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749, pp. 169–180. Springer, Heidelberg (2007)
7. Ghose, A., Koliadis, G.: Pctk: A toolkit for managing business process compliance. In: Proc. of the 2008 International Workshop on Juris-Informatics, JURISIN 2008 (2008)
8. Hinge, K., Ghose, A., Koliadis, G.: Process seer: A tool for semantic effect annotation of business process models. In: IEEE Computer Society Press (ed.) Proc. of the 13th IEEE International EDOC Conference, EDOC 2009 (2009)
9. Podorozhny, R., Lerner, B., Osterweil, L., Podorozhny, R., Lerner, B., Osterweil, L.: Modeling resources for activity coordination and scheduling. In: Proceedings of Coordination 1999, pp. 307–322 (1999)
10. Kwan, M., Balasubramanian, P.: Adding workflow analysis techniques to the IS development toolkit. In: Proceedings of the Hawii International Conference on System Science, vol. 31, pp. 312–321 (1998)
11. Russell, N., Ter Hofstede, A., Edmond, D., van der Aalst, W.: Workflow resource patterns: Identification, representation and tool support. In: Pastor, Ó., Falcão e Cunha, J. (eds.) CAiSE 2005. LNCS, vol. 3520, pp. 13–17. Springer, Heidelberg (2005)
12. Muehlen, M.: Resource modeling in workflow applications. In: Proceedings of the 1999 Workflow Management Conference, pp. 137–153 (1999)
13. Muehlen, M.: Organizational management in workflow applications–issues and perspectives. Information Technology and Management 5(3), 271–291 (2004)
14. Bistarelli, S., Montanari, U., Rossi, F.: Semiring-based constraint satisfaction and optimization. Journal of the ACM (JACM) 44(2), 236 (1997)
15. Yu, E., Mylopoulos, J.: From ER to 'AR'-modelling strategic actor relationships for business process reengineering. International Journal of Cooperative Information Systems 4, 125–144 (1995)
16. Van Lamsweerde, A., et al.: Goal-oriented requirements engineering: A guided tour. In: Proceedings of the 5th IEEE International Symposium on Requirements Engineering, Toronto, Canada, vol. 249, p. 263 (2001)
17. Koliadis, G., Ghose, A.K.: 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)
18. Weber, B., Rinderle, S., Reichert, M.: Change patterns and change support features in process-aware information systems. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) CAiSE 2007 and WES 2007. LNCS, vol. 4495, pp. 574–588. Springer, Heidelberg (2007)

Towards a Service-Oriented Energy Market: Current State and Trend

Giuliano Andrea Pagani and Marco Aiello

Distributed Systems Group
Johann Bernoulli Institute
University of Groningen
Nijenborgh 9, 9747 AG, The Netherlands
{g.a.pagani,m.aiello}@rug.nl

Abstract. The energy sector, which has traditionally been an oligarchic closed one, is undergoing major changes at all levels: more and more players are authorized to produce, deal and transport energy, and energy consumers are now in the position to also produce and trade energy. This new trend can be supported by Service-Oriented Architectures (SOAs) at all levels. In this short position paper, we overview the current situation of the energy sector and we indicate challenges for SOA to be addressed for a successful unbundling of the energy arena, thus providing a more efficient infrastructure with both environmental and economic benefits.

Keywords: SOAs in practice, Energy, Power Grid.

1 Introduction

The energy production and distribution sector was traditionally a strong monopoly with a fixed hierarchy that resembles a client-server architecture: one large provider and infrastructure owner, and millions of end-users. After decades of very little change, in recent years new regulations, technologies and demands from the public are re-shaping the sector. Governments are pushing for the unbundling of the market (e.g., [2]), renewable energy sources are becoming convenient and available both at the industrial and at the residential scale [14], and environmental concerns about energy production and its side-effects on the climate are becoming a strong focus of public attention. Therefore, the vision is that of an Energy sector where any stakeholder can consume and produce energy (a.k.a. *prosumer*) while being entirely free to trade it in an open market. This brings a huge number of autonomous actors that need to interact, sign service-level agreements, conduct trade and provision energy. The future of the Energy sector might be to become a dynamic and open infrastructure which pervades our lives exactly like the Internet does today.

We argue that looking at the current state and trends of the Energy sector, Service-Oriented Architectures will provide a solid foundation to support the new trend, though to be able to do so a number of new challenges will have to be addressed. Our study is not only based on a literature survey, but also

E.M. Maximilien et al. (Eds): ICSOC 2010 Workshops, LNCS 6568, pp. 203–209, 2011.

personal interaction with the large and medium energy players of the Netherlands (TenneT, Enexis, Essent, Itron, Phase-to-phase, RenQi, among others). This paper is industrial providing vision, requirements, and challenges in relation to SOAs. It is a summary of the extended works [11,10] and it is organized as follows. In Section 2, we consider the current situation of the Energy sector. We then provide an overview of technologies used for energy trading in Section 2. Section 3 illustrates the challenges of supporting the future scenario and the relation to SOAs. Concluding remarks are summarized in Section 4.

2 The Energy Sector

The Energy sector is ready for a radical shift driven, on the one hand, by technological innovation with, for example, the introduction of ecological generation facilities (both at a large and micro-scale level) and distributed power sources [8] and, on the other hand, by the political push to break the monopolies and unbundle the market. This is supported, for example, by recent EU directives emphasizing the strategic importance of a Smart Grid approach [5,6]. But first we consider the organization of this sector. The Power Grid is partitioned (both topologically and organizationally) into three major segments: High, Medium and Low voltage. Energy is mainly produced in large facilities at the High voltage level by a few authorized actors while end users exist mostly at the Medium and Low voltage. The structure is hierarchical. The new directives which promote the unbundling aim at placing more actors at the higher levels to improve efficiency and reduce costs while keeping the same level of service and reliability.

The Energy sector is not only about the physical infrastructure for the production and the control of energy transport, but it is also about the data exchanges that have to take place in order to manage energy billing, trading and the business involved in the creation of added value around energy delivery.

The entire system can be divided in three layers that have different purposes:

- **The physical layer** is the lowest in the stack and it interacts directly with the electrical apparatus (e.g., transformers, switchgear, relays) that belongs to a distribution substation and power plant control equipment.
- **The data layer** spans the control data used to supervise and actuate the physical equipment to all the interactions necessary to properly govern the different systems involved in production, transmission and distribution of energy. It enables remote operations on the physical equipment and interaction between the various components of the grid (e.g., sensors and actuators).
- **The business layer** is formed by information coming from the data layer and is a key element for running the business of electricity. It can be used to measure company performances through key performance indicators, to create new business models and opportunities and to make important forecasts for future trading opportunities and needs.

Stakeholders in the Energy Sector. In the current situation, the typical roles an energy company has in a monopolistic market are: power generation,

grid management, energy supply, metering and billing. Previously all these roles were held by only one institution, possibly governmental. The current and future trend is to unbundle the market and increase the number of actors at all levels.

What becomes interesting in the future scenario is that there are several companies involved in the same business segment and all need to send and receive essential information to properly operate. Many players for each business category are able to join and contribute to the flow of information. In order to successfully interact with each other a standard, scalable and secure way of exchanging information is required.

A new Grid. An investigation of the Smart Grid concept with all its facets is beyond the scope of this short paper. However it is important to highlight the potential benefits for energy savings and environment a pervasive installation of Smart Grid technology can provide: savings around 20%. The figure is what the Pacific Northwest National Laboratory has found in a study analysing the energy trends up to 2030 in the U.S.A. [9]. Another study from Electrical Power Research Institute estimates a reduction up to 211 million metric tons of CO_2 per year in 2030 thanks to the savings possible through Smart Grid [4].

What is relevant is the implication in terms of necessary modernization, i.e., the increase in the number of actors and system components that communicate with each other from an information infrastructure perspective (e.g., smart meters, home appliances, plug-in hybrid electric vehicles) and interact directly or indirectly with the energy market.

Current systems for energy trading. Consider six major energy markets (US, UK, Japan, Germany, France, and Italy) which all have an operational

Table 1. G6 Countries Information Systems Energy Markets Interaction Summary

Market Manager	Land	# Participants	Web Interface	Upload Download	Web Service	SOA	Open Solution
GME (http://www.mercatoelettrico.org/)	ITA	190	✔	✔ XML format	✔	✔	Open
ISO New England (http://www.iso-ne.com/)	USA	422	✔	✔ CSV and XML format	✘	✘	N/A
PJM Interconnection (http://www.pjm.com)	USA	612	✔	✔ XML format	✔	✔	Software tools are proprietary, but interfaces are open
Elexon (http://www.elexon.co.uk)	UK	226	✔	✔ format according IDD rules	✘	✘	Open to some extent
European Energy Exchange (http://www.eex.com/en/)	D	160	✔	✔ CSV format (for ComTrader platform)	✘	✘	Proprietary Xetra based software
European Power Exchange Spot (http://www.epexspot.com/en/)	F	80	✔	N/A	✘	✘	Proprietary software (SAPRI, Global Vision)
Japan Power Exchange Spot (http://www.jepx.org/)	JP	48	✔	N/A	N/A	N/A	Open as far the small amount of information suggest

structure very similar to each other providing the possibility to trade energy with different time granularity (e.g., from hours to minutes) and with different time horizons (e.g., from real-time markets to one year forward contracts). In Table 1, we summarize the findings of a comparative study of these markets (full study in [10]). A number of interesting facts emerge from the comparison: (1) the number of participants involved is low and can change considerably; (2) web interfaces are the only constant, while there is varying IT support for interaction; (3) the solutions are mostly open, though not standardized.

3 Open Power Grids via Service-Oriented Architectures

After the legal unbundling of the markets, the appearance of renewable generating facilities at all scale levels, the standardization of the control elements of the Power Grid, and the diffusion of digital/data prone smart meters, all the ingredients seem to be there, but how can these elements provide for a different energy landscape? The challenges that seem to emerge are the following ones.

Interoperation. The number of actors populating the energy market is constantly increasing and their capabilities as well. There is a strong need to have standards for interoperation at all levels (not only the control layer of the grid). Furthermore, standards tend to cover the syntactic part of the interoperation, while the semantics of the message exchange is scarcely addressed.

Scalability. The increase of actors also involves scalability issues. If millions of micro energy producers start trading micro-quantities of energy, there must be an appropriate infrastructure to manage this, possibly real-time, information exchange.

Discovery. If the actors increase and more entities can take on the same role, one may think of discovering services on the fly. The idea of signing an yearly contract for a home, may be too limiting and one may want to switch energy supplier on a much shorter time frame. Furthermore, if anybody can be a supplier, then one may want to find a provider in the instant it is needed.

Mobility. In the future grid energy consumers, and also producers, may be mobile on the grid. Cars will be electric, but may also have energy producing and storing facilities (e.g., a solar cell roof, fuel cells powered engine). The mobile elements need to interact with the Power Grid in a transparent way.

Pervasiveness. Computation needs to be available everywhere on the Grid, if all actors need to participate into the Energy SOA.

Resilience to failure and trust. The electrical Power Grid is a critical infrastructure. A key performance indicator of the current energy distributors is the down time that should not exceed the few hours per year. When moving to an open Smart Grid the delivery of energy must not decrease in quality. This requires having a trust mechanism among the various players and an overall secure infrastructure. It may also require having reliable forecasting of generation and use.

Service Integration and composition. The physical layer, the data layer and the business layer will have to interact more closely as any node who produces energy needs to control it and guide it into the grid, get paid for it, or make the generation part of a larger business process relying on the energy (e.g., one could drive an electric car while lodging at a motel, plug it into the grid and use the car generated electricity to pay part of the bill [14]).

Topology. The current infrastructure is strongly hierarchical. Very few large energy producers and backbone owners, some actors in the middle segment, and many end users with limited flexibility. But the new vision of the open grid demands for a flat peer-to-peer network in which all actors are producers and consumers of energy, data and services.

Real-time. Energy related operation such as control, actuation, distribution and trading have very strict time-dependant constraints to satisfy. All the players in the next generation grid must interact following real-time commitments to provide and receive an energy service with the proper quality.

Interestingly, these challenges are typically best addressed by a Service-Oriented Architecture (e.g., [12]). Let's consider them one by one.

Interoperation. Web services are a technology to build service oriented architecture and address the problem of interoperation being standardized XML protocols to describe messages, remote operations and coordination among loosely coupled entities, e.g., [7].

Scalability. The basic SOA pattern: publish–find–bind allows to decouple service consumers from producers and to substitute, even at run-time, one component for another one. The communication, most often asynchronous, provides all the ingredients for a highly scalable infrastructure. Examples of which have already appeared in the area of eBusiness.

Discovery. Discovery is one of the basic ingredients of a SOA. It needs to support the publish and find operations and is usually based on registries, but can also be realized with flooding models.

Mobility. An SOA supports actor loosely coupling and behaviour based binding, therefore the mobility of the elements is easily supported, e.g., [1].

Pervasiveness. In the SOA paradigm the smart meter is basically a service provider and a service consumer at the same time. It invokes other services to interact on the market and also provide services both to other market participants interested in energy purchase and to intelligent home appliances that require energy at a certain time.

Resilience to failure and trust. Protocols exist to enable a web service based SOA with trust, privacy and security support. This can provide the basic for a secure infrastructure. Reliability and physical security will also have to be pursued with appropriate energy technology which is beyond the SOA.

Service Integration and composition. Service integration and composition is the key added value of an SOA and many examples exist on methodologies to support this, e.g., [3]

Topology. SOAs support any kind of topology. The hierarchical client-server one is less common, but can be realized; P2P is the most common.

Real-time. Solutions are available to introduce enhancements to SOA paradigm in order to provide an appropriate quality of service and satisfy real-time constraints, e.g., [13].

4 Concluding Remarks

The Energy sector is undergoing important changes that are bringing many more players onto the Energy scene. With decentralized small generators everybody can become an energy producer and therefore interact with the market to sell and buy energy. This vision is going to be achieved with technologies that are in development following the Smart Grid vision. More flexible and scalable software solutions based on SOA have already been proposed for control purposes of energy systems belonging to the Smart Grid. The next step is to provide a general infrastructure to support standardized, real-time, open energy trading.

Acknowledgements

Pagani is supported by the Ubbo Emmius Fellowship 2009 provided by the University of Groningen. The work is supported by the EU FP7 Project GreenerBuildings, contract no. 258888.

References

1. Aiello, M., Dustdar, S.: A domotic infrastructure based on the web service stack. Pervasive and Mobile Computing 4(4), 506–525 (2008)
2. Cossent, R., Gómez, T., Frías, P.: Towards a future with large penetration of distributed generation: Is the current regulation of electricity distribution ready? regulatory recommendations under a european perspective. Energy Policy 37(3), 1145–1155 (2009)
3. Dustdar, S., Schreiner, W.: A survey on web services composition. Int. J. Web Grid Serv. 1(1), 1–30 (2005)
4. EPRI: The green grid: Energy savings and carbon emissions reductions enabled by a smart grid. Tech. rep., EPRI (2008)
5. EU: Vision and Strategy for Europe's Electricity Networks of the Future, European Technology Platform SmartGrids. Tech. Rep. EUR 22040, EU (2006)
6. EU: Towards smart power networks, lessons learned from european research fp5 projects. Tech. Rep. EUR 21970, European Commission (2007)
7. Leymann, F., Roller, D., Schmidt, M.T.: Web services and business process management. IBM Systems Journal 41(2), 198–211 (2002)
8. Lovins, A.B., Datta, E.K., Feiler, T., Rabago, K.R., Swisher, J.N., Lehmann, A., Wicker, K.: Small is profitable: the hidden economic benefits of making electrical resources the right size. Rocky Mountain Institute (2002)
9. Pacific Northwest National Laboratory: The smart grid: An estimation of the energy and co_2 benefits. Tech. rep., Pacific Northwest National Laboratory (2010)

10. Pagani, G.A., Aiello, M.: Energy market trading systems in g6 countries. Tech. Rep. JBI preprint 2010-6-01, JBI, University of Groningen (2010)
11. Pagani, G.A., Aiello, M.: Towards a service-oriented energy market: Current state and trend. Tech. Rep. JBI preprint 2010-6-02, JBI, University of Groningen (2010)
12. Papazoglou, M.P., Georgakopoulos, D.: Service-oriented computing. Commun. ACM 46(10), 24–28 (2003)
13. Tsai, W., Lee, Y.h., Cao, Z., Chen, Y., Xiao, B.: RTSOA: Real-Time Service-Oriented Architecture. In: 2006 Second IEEE Int. Sym. Service-Oriented System Engineering (SOSE 2006), pp. 49–56 (October 2006)
14. Vaitheeswaran, V.: Power to the People. Earthscan (2005)

Introduction to the Second International Workshop on Service Oriented Computing in Logistics (SOC-LOG 2010)

Joerg Leukel[1,*], André Ludwig[2,**], and Alex Norta[3]

[1] Department of Information Systems 2, University of Hohenheim, Stuttgart, Germany
joerg.leukel@uni-hohenheim.de
[2] Information Systems Institute, University of Leipzig, Germany
ludwig@wifa.uni-leipzig.de
[3] Department of Computer Science, University of Helsinki, Finland
alexander.norta@cs.helsinki.fi

Abstract. The purpose of the SOC-LOG workshop was to present and discuss recent significant developments at the intersection of service-oriented computing and logistics systems/supply chain management, and to promote cross-fertilization and exchange of ideas and techniques between these fields. The relation to ICSOC 2010 is that, on one hand, the conference addresses the core concepts such as interacting business processes, service composition, service operations, and quality of services, and on the other hand, would receive feedback, experiences, and requirements from a highly relevant application domain to validate and advance its current approaches. The technical program consisted of 4 full papers, 1 position paper (all being reviewed, with an acceptance rate of 56%), and 1 invited talk.

Keywords: logistics, service oriented computing, supply chain management.

1 Aims and Scope

Logistics is of paramount importance for many industries: It plans and realizes the flow of goods from sources to destinations by means of transformations in space, time, and quantity. Coordinating logistics activities faces organizational and technical boundaries of the participating firms as well as must resolve conflicting goals and strategies of such firms. Information plays a crucial role in logistics, in particular in today's business environment, which is changing significantly due to, e.g., globalization of supply chains, stronger customer orientation and individualization, all increasing the need for more adaptive logistics systems. IT is the key enabler for managing these challenges, supporting supply chain collaboration, and managing increasing economic dynamics. Advanced IT support allows supply chains to increase their

* Funding received by the German Ministry of Education and Research under the project InterLogGrid (BMBF 01IG09010E).
** Funding received by the German Ministry of Education and Research under the projects InterLogGrid (BMBF 01IG09010F) and Logistics Service Bus (BMBF 03IP504).

E.M. Maximilien et al. (Eds): ICSOC 2010 Workshops, LNCS 6568, pp. 210–212, 2011.

efficiency significantly, to better fulfill customer needs, and handle the growing organizational complexity and the associated supply chain risks.

While existing logistics IT systems provide solid support for static, self-contained logistics systems, the research on managing the logistics in supply chains that are dynamically changing, is still less advanced. Service oriented computing (SOC) is a promising paradigm, which automates inter-organizational processes with loosely coupled software-based services. The focus of this workshop is the study and exploration of the SOC's potential to solve coordination problems in logistics systems and supply chains.

Logistics systems and supply chains have been subject to much research in Computer Science, Information Systems, and most recently Service Science. Of particular interest are works on adopting ideas, concepts, models, and methods from software technology which follows the service paradigm, i.e., Grid Computing, Service-oriented Computing, and Cloud Computing. One can identify two major research streams: The first stream is adopting the service paradigm for logistics and supply chain management software. This approach aims at providing the functionality by an integrated set of (Web) services. The second stream is adopting the service paradigm for representing logistics systems and supply chain – or parts of it – by (Web) services. In this sense, the electronic representations become the subject of Web service coordination and its respective body of knowledge.

Key research questions are: (1) How to represent logistics systems in service-based computing systems by employing and adopting constructs, models, and methods of the SOC technology stack, (2) how to describe software-based logistics services with service description languages, (3) how to coordinate software-based logistics services, by employing and adopting approaches for service discovery and service composition, (4) how to negotiate and agree about the delivery of software-based logistics services with approaches for SLA representation, SLA management, and SLA negotiation, and (5) how to control the delivery and how to measure the efficiency and effectiveness of software-based logistics services? With the set of design principles, architectural models and concepts and last but not least with its existing and growing set of standards, SOC promotes adaptiveness of logistics systems and supply chains, a flexible and reconfigurable provisioning along multiple supply chains, and their efficiency.

All submissions received were single-blind peer reviewed by at least two and up to four members of the international program committee. In total, we received nine submissions from three countries. Based on the review reports, we accepted four full papers and one position paper, which amount to an acceptance rate of 55.6%. We would like to thank the program committee members and authors for all of their hard work and participation in the lively workshop. We hope that SOC-LOG will help exchanging new ideas and networking and sharing ideas. More information on SOC-LOG 2010 is available at http://soclog10.wifa.uni-leipzig.de.

2 Workshop Co-chairs

Joerg Leukel (University of Hohenheim, Germany)
André Ludwig (University of Leipzig, Germany)
Alex Norta (University of Helsinki, Finland)

3 Program Committee

Witold Abramowicz (Poznan University of Economics, Poland)
Samuil Angelov (TU-Eindhoven, Netherlands)
Marcelo Cataldo (Carnegie Mellon University, USA)
Rik Eshuis (TU-Eindhoven, Netherlands)
Diogo Ferreira (IST – Technical University of Lisbon, Portugal)
Paul Grefen (TU-Eindhoven, Netherlands)
Maria-Eugenia Iacob (University of Twente, Netherlands)
Axel Korthaus (Queensland University of Technology, Australia)
Marek Kowalkiewicz (SAP Research, Australia)
Carlos Müller (University of Sevilla, Spain)
Manuel Resinas (University of Seville, Spain)
Manfred Reichert (Ulm University, Germany)
Toni Ruokolainen (University of Helsinki, Finland)
Jun Shen (University of Wollongong, Australia)
Vijay Sugumaran (Oakland University, USA)
Ingo Weber (University of New South Wales, Australia)

Coordinating Distributed Operations

Daniel Oppenheim, Saeed Bagheri, Krishna Ratakonda, and Yi-Min Chee

IBM Thomas J. Watson Research Center, Hawthorne, NY 10562, USA
{music,sbagher,ratakond,ymchee}@us.ibm.com

Abstract. In this manuscript, we discuss the need for, and present a new system architecture for cross collaboration among multiple service enterprises. We demonstrate the importance and inevitability of such collaboration along with challenges in its proper realization through several real-life examples taken from different business domains. We then show that these challenge are rooted in two key factors: unpredictability and responsiveness. The key contribution of this manuscript is the presentation of a new model, centered on an intelligent hub, for coordinating the logistics of cross enterprise collaboration. This hub is constructed in a manner intended to directly identify and solve the two key fundamental challenges of cross enterprise collaboration. As such, we expect it to outperform other means of collaboration across service providers. We demonstrate the potential for such performance using field examples.

Keywords: service operation management, service quality, logistics, enterprise ecosystem, globalization.

1 Introduction

Service operation and management is becoming increasingly complex as the *doing* of work shifts from within the enterprise to a distributed ecosystem of service providers. While this new ecosystem provides a desirable flexibility [1,2], it challenges both the sustained effectiveness of operations and the quality of delivered services. It has accordingly become apparent that a new approach is required to address this challenge [3]. Using such an approach service operation management will become *aware* of the ecosystem and have the ability to optimally manage the logistics of its inherent complexity. Moreover, a business could then manage its operations while reasoning within its own business terms. Such optimal operation management will allow us to realize the potential benefits of the ecosystem while maintaining an acceptable level of quality.

Forces of globalization and the ever growing need for differentiation and innovation are moving work from a co-located to a distributed environment. Companies form collaborations to achieve a goal that none could achieve individually. They all share the financial burden and development risk, but benefit from the sum of their differentiating capabilities. We call this new model of doing business Cross Enterprise Collaboration (CeC). For example, Airbus is developing its new Airbus-A380 in a consortium with several partners, and similar trends

E.M. Maximilien et al. (Eds): ICSOC 2010 Workshops, LNCS 6568, pp. 213–224, 2011.

can be observed in all industries. However, collaboration, while inevitable, is coming at a huge price. The rate of project failures is alarming. The delivery of the Airbus-380 was two years overdue with accumulated losses exceeding 6B USD [4]. This breakdown, while extreme, is common across different industries, including software and automotive. Clearly, there is a need for a new approach to manage cross enterprise collaboration.

The problems of CeC are well recognized by industry. A study made between 2007 and 2008 by the Aberdeen Group [5] examined more than 160 enterprise products that required collaboration between mechanical engineering, electronic engineering, computer engineering, control engineering, and system design. 71% of the people interviewed identified a need to improve the communication and collaboration across the silo disciplines; 49% required improve visibility; 43% raised the need to implement or alter new product development processes for a multidisciplinary approach. The primary reason for breakdowns is the paucity of support for coordinating the overall work between the silo disciplines.

We approach the problem from two complementing directions. First, we model the ecosystem of the enterprise collaboration. This provides the context in which a business can identify the main actors and reason about how they relate to each other: organizations, people, process, and work. Second, we use service oriented concepts to encapsulate the *doing* of work from managing and *coordinating* how it gets done. This encapsulates each service provider in a space separate from the rest of the ecosystem. Accordingly, only the necessary and sufficient information of the ecosystem needs to be communicated in order to coordinate a provider. Once this is done, the complexity of Business Process Management (BPM) suddenly reduces to many independent sub-BPM problems that are both known and tractable. The challenge, however, is formation of the appropriate encapsulated space for each service provider and definition of the information flow. Another way to think of this encapsulation is as an extension of the concept of Work as a Service (Waas), where any domain work be provided as a service and carried out by any qualified provider organization. A key component of the model is dedicated specifically to support this: a hub. In a companion paper we provide detailed analysis of hub operation and show how an optimum policy can be determined. This policy, among other things, describes how service requests are distributed amongst providers [6]. The model we present here presents a broader view and addresses more directly business needs and concerns. The model might be viewed as a significant generalization of the Distributed Enterprise Services pattern [7,8], and incorporates Malones theory of coordination [9] and work [10].

In the next section we illustrate the core challenges facing a proper implementation of CeC. In particular, we discuss the role of unpredictability and how it affects the complexity of CeC. We further introduce a new perspective for decision making that is both holistic and domain aware. In section 3 we incorporate our new perspective in decision making into the structure of an intelligent hub and propose a new model to support CeC. We demonstrate the operation of our model through an illustrative example. The section after that reviews the key properties of our hub model such as agility. We conclude in section 5.

2 Collaborative Enterprise Work in Development

Product development is complex and is carried out in large projects. Projects require deep domain expertise in several disciplines, last from months to years, and are expensive; their success is critical to the business. Detailed processes, methodologies, and best practices guide the project lifecycle. However, plans can only form a common starting point because unpredictable events always happen. Issues may arise following a realization that a requirement is unclear or a design is faulty. Assumptions may prove to be wrong. Required materials may become unavailable or new technologies may make current progress obsolete. Partners or providers may come and go, and disasters certainly happen. Uncertainty and unpredictability can only be managed by flexibility and adaptation, which is why every project deviates from its original plan in unique ways.

Successful completion of development projects depends on containing their inherent unpredictability. To do this one must identify issues early, figure out the best response, and make the necessary changes. Containment in CeC work is too complex to automate. Were Airbus able to automate the creation of a blueprint for the A-380, it would. CeC work, therefore, has a critical dependency on human knowledge, experience, leadership, and creativity. People must do everything that technology cannot, and technology must support them in doing what is left. To understand what needs to be supported and how, we next examine the different roles of humans and organizations in responding to an unpredictable event.

2.1 The Three Perspectives: Executive, Operation, and Domain

Figure 1 depicts the three fundamental roles of people: executive, operation, and domain. The *executive* role is responsible for the enterprises strategy and business goals; it also pays close attention to the marketplace, competitors and customers. *Operation* is a managerial role that is concerned with efficiency, and meeting deadlines; it focuses on the coordination of work between the silo disciplines. *Domain* roles embody the competency and skills required to do the highly specialized work that is the underpinnings of the final product.

2.2 Containing Unpredictability through Exception Handling

Determining the best response to an unexpected event can be thought of more generally as *human exception handling*. The following scenario illustrates how the three roles relate to each other in the context of CeC work. Large development projects distribute work between several teams that do *requirements, design, build, test,* and *integrate*. In this scenario, the electric design team realizes that it may be two month overdue. It wants to determine how serious the problem is so that it can decide what to do about it.

The seriousness of the problem can range anywhere from having no impact at all on the project to killing the enterprise by giving a competitor a two month advantage. The team cannot possibly determine which because it lacks visibility into the operation and executive domains.

What is a reasonable response? The team may decide to reallocate resources from other tasks in order to overcome the anticipated lateness. However, this may be the worst thing to do. For example, it may turn out that the root cause stems from an engineering error in the engine's design. If such is the case, the engineering team should first fix its design before anyone takes further action, as the error may affect all ongoing and future work. Operations, on the other hand, have different concerns and will likely make a different decision. If, for example, meeting the project deadline is a key priority, then operations may insist on meeting the original deadline, well aware that quality will be seriously sacrificed. On the other hand, an executive perspective may lead to a very different approach: termination of the entire project at a loss and reallocation of its resources to a more profitable venture.

All the approaches described above are reasonable, but each reflects a different concern. To make an informed decision one must have the full context and balance all related concerns. Figure 1 depicts the creation of the required context by gathering information and concerns from each level. Information about the domain can be gathered from each discipline. Operations can provide relevant information relating to the project scope. The partnering executives can add relevant information from the perspective of goals, strategy, competitors, or the marketplace. But there may be additional information that is required: new technologies, advancements by a competitor, etc. This is why key stakeholders must take part in the decision making process: to voice their concerns and supply additional information and insight that can only come from human experience and expertise.

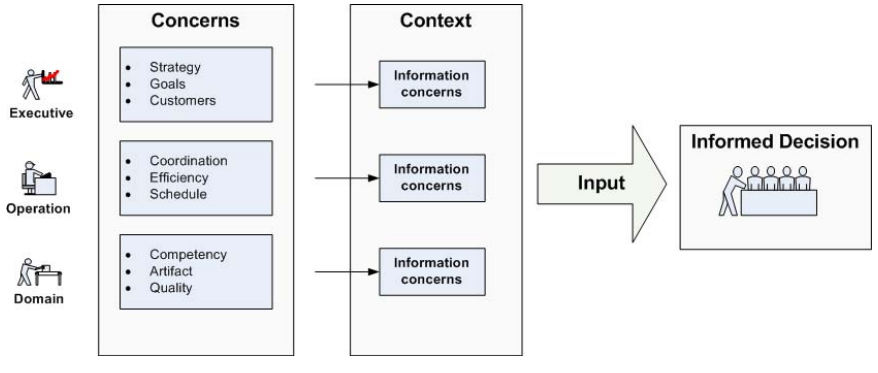


Fig. 1. When something unpredictable happens it must be contained. The sum of information that is gathered from all three domains provides a big picture. Based on this context the stakeholders can make an informed decision.

2.3 Patterns of Coordination

Figure 1 outlines a desirable *issue resolution* process. To enact that which was decided upon requires coordination. But coordination can be complex. If the root problem of the electric design team was a fault in an engineering design, then the need to fix the design can easily be communicated to engineering. However, other teams may also have dependencies on engineering and they must also be notified. So far, so good, but this is where complexity begins to compound. The changes that engineering make may require other disciplines to modify their designs so that they accommodate the changes which engineering made. Each new change in every other discipline may start a new cycle of updates and dependencies between all disciplines. This situation can quickly become unmanageable.

Figure 2 depicts two patterns of coordination: *collaborative* and *centralized*. Collaborative collaboration is a remnant of co-location; it lacks an entity dedicated to coordination. It considers all participants as equal partners, as at first seems appropriate in an ecosystem of co-producers. In this pattern, which is current practice, every team will start communicating with every other team to try and figure out the root cause for its problem and the best thing to do about it. There are, however, two fundamental drawbacks. First, because the operation and executive scope are missing, they are unlikely to make the *right* decision, as described in Fig. 1. Second, the extremely wide bandwidth of communication is both inefficient and distracting.

In the *centralized* collaboration pattern coordination is handled by one entity comprising both operation and executive. The electric design team will communicate its problem to one, and only one, entity. Because the perspectives of all three roles are now involved, the entire context is available to support an informed decision as illustrated in Fig. 1. Accordingly, individual teams will be coordinated by operations only when needed. Additional changes that are side products of the first will be managed and coordinated in the same way. The centralized pattern supports both making the best decision and the most effective way to coordinate change. It also minimizes the need for cross silo collaboration.

2.4 Communication and Coordination across Silo Disciplines

We can now pose a critical question: what information must be communicated between operation and each discipline in Fig. 2? The simplicity of the answer may seem counter intuitive: the discipline must receive all the information it requires to do its work; operations must satisfy their need to monitor progress so that they can both continue to detect unpredictable events and contain them.

The principles outlined above are used to formulate the basic encapsulation of our model as is described in the following section. The centralized pattern represents an encapsulation of coordination work between autonomous entities. Each discipline in the centralized pattern described in Fig. 3 represents an autonomous entity. The pattern encapsulates each discipline from the provider ecosystem. By doing so it simplifies the problem of coordination from the unmanageable space of an ecosystem to a simple space between two entities. This space is well

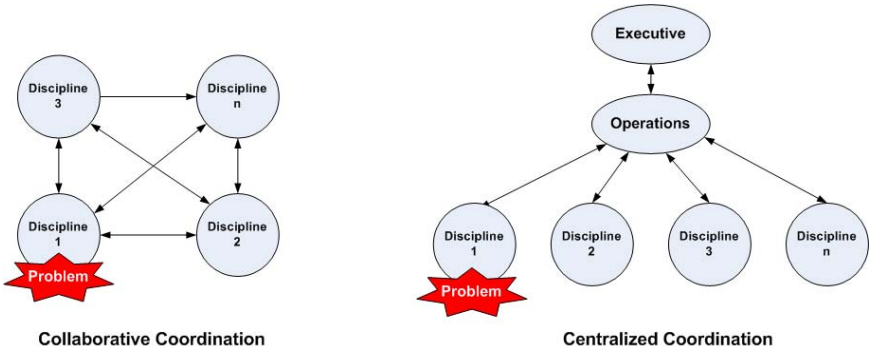


Fig. 2. Collaborative and centralized coordination pattern. Collaborative coordination taxes the communication bandwidth and because it lacks the operation and executive perspectives, is unlikely to yield a satisfactory response. The centralized pattern minimizes the communication bandwidth and lead to good decisions.

understood and can be easily managed. Furthermore, the pattern clarifies what flow of information is required between a single discipline and the coordinating entity.

Figure 3 depicts a proposed standard for Engineering Change Order (ECO) [11]. This process is critical to development and is used to manage ongoing changes. The breakdowns identified by the Aberdeen Group that were mentioned in section 1 are also related to the BPM complexity of coordinating many processes across the silo disciplines. However, as explained above, the execution of the ECO process with one discipline is well understood and can be well handled. The encapsulation provided by our model reduces the the BPM complexity to management of individual processes, such as this ECO, which is well understood and can be easily managed.

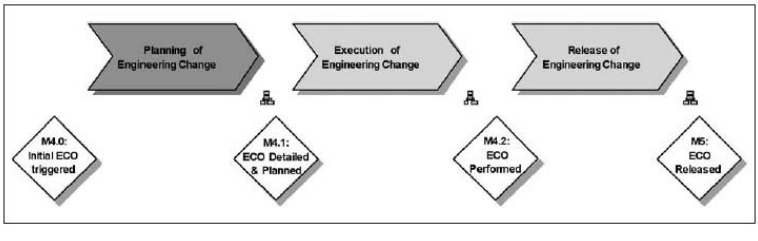


Fig. 3. This is the proposed standard for Engineering Change Order proposed by ProStep [11]. This process also illustrates the information flow between the hub and a provider. To the provider it specifies what it needs to do its work. To the hub it specifies the milestones that enable the hub to monitor progress.

3 A Model of Cross Enterprise Collaboration (CeC)

Our model of CeC work is depicted in Fig. 4. There are three central ideas: a hub that both provides the encapsulation of each organization from the ecosystem and supports the collaboration, the application of the centralized coordination pattern, and the encapsulation of domain work as a service (WaaS).

The basic components required by the hub include organizations, people, and work. In all these components we distinguish between *templates* and *instances*. Templates are designed to be reusable in a wide range of situations. They provide a standard way of doing things in a way that yields predictable results. For example, the ECO described in Figure 4 represents a process template for change management that, being defined as an industry standard, is designed for reuse across any discipline [11]. Instances, on the other hand, are designed to provide the flexibility and agility that are required at runtime. Instances are created only when needed and are then configured in the context that they are needed. This configuration is where specific information, such as requirements, defects, or designs, are specified. Configuration is the mechanism through which templates are optimally adapted to a specific run-time context. However, configuration can continue through the instance lifecycle. This enables ongoing adaptation that is critical for the containment of unpredictability. For example, in response to the problem identified by the electric design team in section 2.2, the hub may decide to modify an ECO instance being serviced by a different discipline, such as engineering. Runtime changes may add defect and design artifacts obtained from the electric team, or even add additional milestones that will verify that the new problem has been appropriately addressed.

Different hubs may modify some components to meet specific business needs. For example, additional organizations can be added to model suppliers, retailers, or customers. Therefore, the model presented here can be thought of as a meta-model that can be tailored to the specific way an enterprise operates. We next describe the necessary and sufficient components that are required by the hub

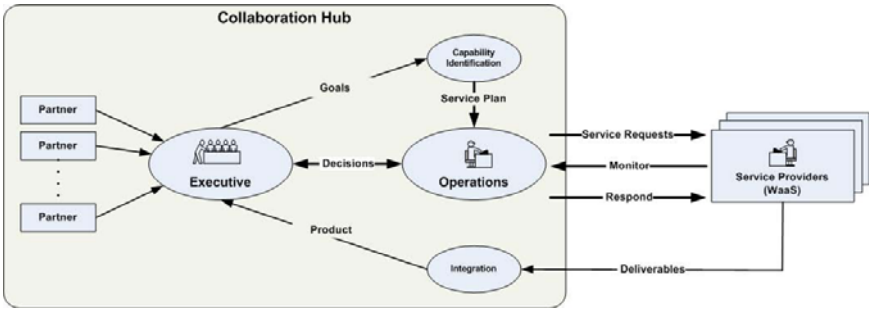


Fig. 4. The hub supports the collaboration across the ecosystem. It simplifies the problem of coordination by encapsulating each provider from the ecosystem. Depicted are the main hub teams that govern the collaboration and the flow of information between the hub and a provider organization.

and then provide an intuitive description of its operation. In the discussion that follows we explain how the model can support agility of operations across the collaboration ecosystem.

People are modeled by templates as roles, skills, and the organization type they belong to. Teams represent instances of people that together perform some common function. For example, a hub has, among others, an executive and operations team. Teams can be virtual, and the number of people in a team can be anything from one and upwards. A service provider has a *coordinator* team. Because a service provider is encapsulated, it own teams within each discipline are not required by the model.

Organizations are differentiated by role. We discuss *partner* and *provider*, but other roles exist, such as *supplier, retailer* or *customer*. An organization has two main attributes: *capabilities* and *roles*. Capabilities represent different types of work an organization can perform, and are explained below. For example, software development disciplines will have the capability to *design, build*, and *test* software. The hub itself, as it represents the virtual organization of partners, will have its own capabilities. For example, *project operation management*, *issue resolution*, or *capability identification*. In addition, each organization type specifies the roles that are required to support the collaboration. Each provider must have a coordinator role. Every hub must have an executive and operation role. Additional roles can be added to any organization.

Work is modeled by templates we call *capabilities* and instances we call *service requests*. The hub provides a fundamental encapsulation for each organization in the collaboration. The *doing* of work, therefore, is always carried out by some organization.

The prime function of capabilities and service requests is to specify the *flow of information* between the hub and a single organization. The necessary and sufficient information is (1) the information required by the organization to do the work and (2) the checks and balances required by the hub to monitor progress and detect issues. To do its work a discipline will require artifacts, such as requirements, use cases, and designs. Specification of process steps on how to do the work are usually not required as it is expected that each discipline follows its own methods and best practices. Checks and balances can include, among other, milestones, metrics, policies, and regulations. This appears quite similar to the ECO standard process shown in Figure 3. However, when a hub is set up, the collaborating partners may modify work templates to reflect their specific way of doing business. Moreover, they may agree on specific policies or regulations that will reflect their approach to governance. Thus, even on the template level, the hub provides a general mechanism for an enterprise to tailor its processes and governance.

Different capabilities represent different disciplines and their specializations. For example, each discipline may have its own capability for *requirements, design, build*, and *test*. Capabilities may be more specific, such as design of *security, GUI*, or *system*. Each capability, therefore, represents a *type* of work that is defined in terms of a discipline and skills that are required to do it. When work is needed,

a capability template is instantiated into a *Service Request*, configured, and then dispatched to a provider. For the provider, configurations will specify information such as schedules, deliverables, and specific artifacts that are required to do the work. For the hub, configurations will include milestones, metrics, and policies.

The complexity of the overall work requires that different parts be carried out by different providers. A *Service Plan* is the construct that both brings together the different capabilities and coordinates the overall doing of work. Because of the provider encapsulation, a service plan is relatively simple from both process definition and management perspectives. This relative simplicity has been well documented by Malone in his theory of coordination [9] and work [10]. In particular, Malone identifies that there are only three basic types of dependencies among activities: flow, sharing, fit (aka *sequence, fork*, and *merge*). Accordingly, a service plan need only order the required capabilities according to their dependencies in sets of sequence and parallel. This is a well understood problem that can be easily managed. The service plan provides a fundamental simplification of process coordination across an ecosystem.

This simplification of process definition implies that creating a service plan is relatively straight forward and can be done quickly. Furthermore, once a plan is created, the instantiation and configuration of each capability is also straightforward and can therefore be done quickly. The time from identifying a need for work to dispatching each service plan to a provider can be relatively short. We therefore introduce the notion of *Just in Time Service Plans*, to highlight the potential for this agility [12].

3.1 Example

We will now illustrate the operation of the hub described in Fig. 4 through an example. In this scenario several companies decide to collaborate on the development of a new car. To support this collaboration they are going to instantiate a collaboration hub. Representatives from each partner join an executive and operation team. Together they define the high level business goals and strategy. They then focus on two things: how they want to govern; and how they want to manage their collaboration. The output of this process will identify two sets of capabilities: domain and hub. Domain capabilities represent different disciplines, such as software or engineering. Capabilities can be defined for *requirements, design, build*, and *test* within a domain; or even be more specific, such as design of *security, GUI*, or *system*. Each domain capability, therefore, represents a type of work which the hub will require.

Hub capabilities will include templates for frequently recurring activities, such as change management, issue resolution, and on boarding providers. For some capabilities, such as change management, industry standards may provide a starting point. The teams may decide to modify the standard so that it better reflects their particular way of doing business. For example, they may require that before any work can be assigned to a provider he must be validated. To do this validation the teams would define a new hub capability called *verify-provider*. It could test for qualifications, require specific IT configurations, or

impose specific governance polices that must be carried out by each provider. The benefit is that this collaboration-specific way of doing things can now be applied generically before any work is assigned to a provider.

The second outcome is identification of additional teams that comprise the hub. In our example, four additional teams were decided upon: *operations, capability identification, integration*, and *issue resolution*. This identification reflects the partners decision to consider these three activities as core to their collaboration. The implications are that the teams will comprise members from each partner, rather than from some provider. The partners themselves will comprise the organizations that provide these capabilities to the hub as a service. Next, the partners would identify specific organizations that will service domain capabilities. This activity could be managed by a hub capability called *onboard provider*. Providers are added to the hubs list of qualified providers.

Now that the hub has been setup, it can begin to operate. The executive team would determine its immediate goals. Based on these goals, the hubs capability identification (CI) team would identify what capabilities are required to achieve these goals. These capabilities would be assembled in a *service plan*. The service plan would be passed to operations. For each capability, operations would identify the optimal provider and configure a service request (a rigorous formulation is provided in [6]). The service request would be dispatched to each provider, who would in turn, do the work. Once the work is completed, the provider would hand over its deliverables to the hubs integration team. When all deliverables from all providers are integrated into the final product, it is delivered to the partners.

So far we have described the flow of service requests and deliverables between the hub and a provider. Another critical aspect of information flow is *monitoring*. This provides the hub with an ability to satisfy itself that the work carried out by each provider is executing according to plan. Refereeing back to the example we gave in section 2.2, the hub would have detected that the electric design team may be two months overdue well in advance. As a result, the hub would automatically instantiate its *issue-resolution* capability. The configured resolution service request would identify the stakeholders from the domain team and its own operation and executive teams, collect the relevant information from the domain, operation, and executive domains, and support them in deciding what to do about as illustrated in Fig. 1. When a decision has been made, modification to ongoing work will be coordinated by the hub with each provider in accordance with the centralized coordination pattern described in Fig. 2. This coordination will be communicated between the hubs operation team and the providers coordination team. If new work is required, the hubs CI team will identify the required capabilities and assemble a service plan. Then, operations will instantiate the plan and dispatch it to the appropriate providers.

4 Discussion

The model provides two main benefits: simplification of the distributed BPM problem and a conceptual framework for a business to reason about its

operations. By encapsulating each provider away from the ecosystem the BPM problem between the hub and each provider is simplified and easily managed. By encapsulating the coordination of work across the ecosystem in a hub, the complex problem of cross enterprise BPM is both bypassed and managed uniformly by a dedicated entity. The hub is a reusable meta-framework that is designed to handle the complexities of coordinating distributed work. It can be tailored to different industries, and instances will reflect the business practices of the collaborating partners. This removes the burden and complexity of needing to support coordination through the choreographies of business processes. This simplification further allows the process designer to focus on the unique business needs which the process is addressing. The design around the centralized collaboration pattern provides a reusable framework to coordinate work and contain unpredictability.

To the business the model provides a conceptual framework for it to reason about its operations. Different industries may adapt the model to their particular needs by adding, for example, customer or supplier organizations. Each business would populate the model according to its own goals, governance, policies, and best-practices. The populated model would represent that particular enterprises knowledge. This knowledge, in turn is used to drive its operations. As experience and insights are gained, they can be incorporated into the model, and immediately become operational.

Consider, for example, the root cause for the Airbus-380 failure: the German and Spanish teams used CATIA version 4, while the British and French teams used CATIA version 5 [13,4]. The collaboration failed because it did not implement a capability to onboard a new provider and check, among other things, for IT incompatibilities. Had this collaboration been supported by a hub, then the hubs *validate-provider* capability would have been used before any work was assigned to any provider. Consider, for example, the ECO in Fig. 4. Before executing any step defined in the ECO, the hub would execute its validate-provider capability. The ECO is an industry standard process. The validate-provider is a collaboration specific process. The model enables a business to tailor an industry standard process to the specific business context of the collaborating partners.

5 Conclusions

In this paper we have introduced a new model for CeC through an intelligent hub. We have shown how our model addresses the key challenges facing proper implementation of CeC: unpredictability and responsiveness. Specifically, by encapsulating each service provider and customer (or any other interacting entity) in its own space and supplying it with necessary and sufficient information, the complex BPM problem for a dynamic ecosystem is reduced to many sub-BPM problems that are known and tractable. Furthermore, our hub is designed to facilitate both the identification and the solving of two root issues of unpredictability and services evolution. The former is achieved through enhanced visibility, metrics, and traceability whereas the latter is achieved through modularity of work and service providers which are all key aspects of the hub design.

Furthermore, we have demonstrated the scalability of this framework, its ability to bypasses the complexity of cross-enterprise BPM, and its flexibility to be tailored to different industries and specific enterprise policies.

References

1. Schmidt, R.: Web services based architectures to support dynamic interorganizational business processes. In: Jeckle, M., Zhang, L.-J. (eds.) ICWS-Europe 2003. LNCS, vol. 2853, pp. 123–136. Springer, Heidelberg (2003)
2. Oppenheim, D., Ratakonda, K., Chee, Y.-M.: Enterprise Oriented Services. In: Dan, A., Gittler, F., Toumani, F. (eds.) ICSOC/ServiceWave 2009. LNCS, vol. 6275, pp. 82–95. Springer, Heidelberg (2010)
3. Grefen, P.: Towards Dynamic Interorganizational Business Process Management. In: Proceedings of the 15th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE 2006 (2006)
4. http://en.wikipedia.org/wiki/Airbus_A380
5. Aberdeen Group., http://www.aberdeen.com/Aberdeen-Library/3365/RA_Mechantronics_CJ_3365.aspx
6. Bagheri, S., Oppenheim, D.: Optimizing Cross Enterprise Collaboration using a Coordination Hub. To be published in the Proceedings of the first Global Conference on Service Research and Innovation (SRII), San Jose, USA (2011)
7. Bhattacharya, K., Caswell, N., Jumaram, S., Nigam, A., Wu, F.: Artifact-centered operational modeling: Lesons from customer engagements. IBM Systems Journal 46(4) (2007)
8. Bhattacharya, K., Hull, R., Su, J.: A Data-Centric Design Methodology for Business Processes. In: Cardoso, J., van der Aalst, W.M.P. (eds.) Handbook of Research on Business Process Modeling. Information Science Publishing, and imprint of IGI Global, Hershey, PA, USA (2009)
9. Malone, T.: What is Coordination? Paper presented at National Science Foundation Coordination Theory Workshop (1988), http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.86.8170&rep=rep1&type=pdf
10. Malone, T.: The Future of Work. Harvard Business School Press, Boston (2004)
11. ProStep: Engineering Change Order (ECO), PSI 3-2 (Draft), Version 0.9. ProStep iVip., http://www.prostep.org/en/downloads/recommendations-standards.html
12. Hull, R.: Private communication (2009)
13. Dassault Systemes, http://en.wikipedia.org/wiki/CATIA

Early Model-Analysis of Logistics Systems*

Freeha Azmat[1], Laura Bocchi[2], and José Luiz Fiadeiro[2]

[1] Department of Engineering, Bahria University, Islamabad, PK
freehaazmat@bahria.edu.pk
[2] Department of Computer Science, University of Leicester, UK
{bocchi,jose}@mcs.le.ac.uk

Abstract. In logistics, as in many other business sectors, service-oriented architectures (SOAs) are offering the possibility for applications to interact with each other across languages and platforms, and for external services to be procured automatically through dedicated middleware components. In this setting, one of the critical business aspects that need to be supported is the negotiation of non-functional quantitative aspects, such as costs, leading to service-level agreements (SLAs) between business parties. In this paper, we present a formal, probabilistic approach through which services can be analysed in relation to the probability that a quality of service property is satisfied.

1 Introduction

Business integration through Web-service infrastructures is becoming prevalent in business sectors that, like Logistics, depend on the procurement of services from external providers to deliver their own services. Service-oriented architectures (SOAs) support a new generation of business processes that can involve outsourced functionalities discovered at run time, automating the procurement of services so as to optimize quantitative quality-of-service (QoS) properties [15] that are essential for their business operation. For this purpose, the dynamic discovery and selection of services needs to rely on the negotiation of service-level agreements (SLAs) between providers and requestors. In this scenario, it is critical that providers be able to guarantee the non-functional properties of the services that they advertise. It is also critical that service requesters can decide which requirements to specify on the services that they need to procure in order to meet given business goals.

To this aim, we propose a formal approach for the quantitative analysis, at the early stages of systems design, of service-oriented applications that allows developers to tune QoS properties offered or required. A number of approaches feature quantitative analysis in the early stages of system design in order to prevent that design choices affect the deployment of the system in a negative way (see [3] for an overview); however they are not specifically targeted to SOAs.

In this paper, we extend the approach that we developed in [6] for the analysis of time-related properties of service models, in order to address a wider range of QoS properties involved in the provision of logistic services. The quality of a logistic

* This work has been partially sponsored by the Leverhulme Trust Award "Tracing Networks".

E.M. Maximilien et al. (Eds): ICSOC 2010 Workshops, LNCS 6568, pp. 225–236, 2011.

system is typically described in terms of a number of different QoS indicators [10,17]. For example, when specifying the model for a logistics cycle (e.g., involving the integration of different functionalities such as information, transportation, inventory, warehousing, material handling, and packaging), one may need to make sure that the cost of the overall business process does not exceed a given threshold, or that the cost of the service being modelled does not exceed the advertised price.

The proposed approach is presented in four steps:

1. We use a simple product-supply service to present a language for modelling services as compositions of internally-provided and outsourced functionalities.
2. We illustrate how the models produced in this language can be annotated with rates that capture cost indicators. In Section 6 we discuss how our approach can be used for other QoS indicators (e.g., *throughput*, *scalability* or *reliability*) that satisfy certain requirements.
3. We show how quantitative analysis of the annotated model can be performed.
4. We show how the feedback obtained from the analysis can be used for improving the original model so that overall non-functional constraints can be guaranteed.

The modelling language that we use in step (1) is SRML – the Sensoria Reference Modelling Language [9] – which was developed in order to capture and formalize foundational aspects of SOAs, including service composition, dynamic binding and reconfiguration, and service level agreements. SRML follows the basic principles and structures put forward by SCA [20], a middleware-independent framework proposed by an industrial consortium for building business applications over SOAs. Whereas SCA focuses on the assembly and deployment of service-oriented software artefacts, SRML offers high-level primitives for business modelling, from component interoperability to business integration.

As a case study, we consider a logistics service for supplying products of a certain kind that finds the warehouse closest to the customer if the chosen product is not available in a local stock [11]. An example of an SLA constraint that we would like to certify is whether in 85% of the cases the modelled business process is able to maintain the costs of the transaction lower than, say, $6. As could be expected, the overall cost depends on the costs of the services that may be discovered at run-time.

In step (2), we annotate SRML models with rates representing a non-functional property. The resulting models are then encoded into the Performance Evaluation Process Algebra (PEPA) [12] so that they can be analysed. PEPA has a formal semantics and is supported by a range of powerful analysis tools [22]. Consistency properties of the encoding have been proved in [6] that show that the encodings preserve the structure of the original SRML models which, together with the fact that the encoding is defined in a modular way, makes it straightforward to establish a correspondence between interactions in PEPA models and events in SRML models. This is particularly useful for obtaining immediate feedback on the original model from the quantitative analysis of the PEPA encoding.

1.1 Related Work

Quantitative model analysis. In [1] the authors propose an approach for stochastic analysis of logistic models based on Petri-nets. SRML supports a higher level of

abstraction and offers domain-specific primitives for modelling business processes. In [13,14] Iacon and Jonkers propose a layered analysis approach covering technical infrastructures, software applications, business processes and products; they focus on performance analysis taking into account inter-relationships between the different layers. Our approach considers two layers – architectural and behavioural – and focuses on properties that arise from the dynamic aspects of service-oriented systems, namely discovery and loose coupling, which are key for flexible architectures.

Analysis of logistic systems. [7] compares agent-based and equation-based approaches to the modelling of logistic services, and [24] uses an equational model to analyse the adjustment of costs in logistic systems. Our approach proceeds top-down by first creating a high-level model and then deriving a corresponding lower-level PEPA (i.e., equation-based) system. Due to the modularity of the encoding (see Section 4) the feedback obtained from the PEPA system can be straightforwardly applied to the re-engineering of the high-level model.

QoS description and run-monitoring of conformance. [8,23] present methods for the evaluation of the best match among the available logistic services considering multiple properties that are possibly related with each other. [25] presents a semantic description model for QoS and proposes a solution for dynamic assessment, management and ranking of QoS values based on user feedback. [16] proposes a method for service evaluation where reputation is a function of user ratings, quality compliance and verity (i.e., the changes of service quality conformance over time). In [18] services are rated in terms of their quality, with QoS information provided by monitoring services and users. [19,21] use mainly third-party service brokers or specialized monitoring agents to collect performance of all available services in registries, which would be expensive in reality. Instead, our approach focuses in analysing that such properties are consistent with a model. In addition, we address quantitative analysis at the early stages of system design in order to prevent that design choices affect the deployment of the system in a negative way.

2 Modelling a Supply Service with SRML

The unit of design in SRML is the *module*, through which business processes are defined as orchestrations of tightly-coupled components that are locally implemented and loosely-coupled dynamically-discovered services. Figure 1 illustrates the structure of the module *Product Supply* that, according to customer preferences, checks for the availability of a product in a local stock or outsources it from an external warehouse that is discovered and selected according to given SLAs. The module includes: (1) a *provides-interface CR* describing the interface that the service offers to customers; (2) a *uses-interface ST* describing the interface of a local database that keeps record of the products in stock; (3) two *requires-interfaces*, *DL* and *WR*, that specify the interfaces to services that may need to be procured on the fly from external parties – a delivery service and a warehouse, respectively; (4) a *component-interface OS* describing the interface of the component that orchestrates the parties that, together, deliver the service; and (5) a number of *wires*, *CO*, *OT OD* and *OW*, that specify interaction protocols between those parties. Whereas *ST* is part of the resources that are

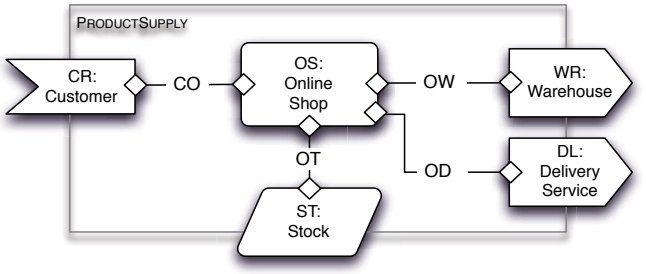


Fig. 1. SRML Module structure of service *Product Supply*

local to *Product Supply*, *WR* and *DL* are discovered at run time only if and when they are necessary (when the product is not in stock) depending on information that is not known a-priori such as the specific product that the customer needs, the quantity, the location for delivery, and so on.

Formally, a SRML module *M* consists of

- A set *nodes(M),* which is the union of the following four disjoint sets:
  - *provides*(M) consisting of a single provides-interface,
  - *uses*(M) consisting of uses-interfaces,
  - *requires*(M) consisting of requires-interfaces,
  - *components*(M) consisting of component-interfaces,
- A set *edges(M)* consisting of (*wire-interfaces*) where each edge *w* is a set of two nodes $w: m \leftrightarrow n$.
- *A* labelling function $label_M$ that associates a behavioural specification with every node, as discussed in Section 2.1.
- An internal configuration policy that specifies initialisation conditions for component-interfaces and triggers for the discovery of external services.
- An external configuration policy *cs(M)* that describes the SLA properties of *M*, as discussed in Section 2.2.

2.1 Modelling the Behaviour of Services

The behavioural properties of the service modelled through a module results from the specifications of all its nodes and wires. Each specification consists of a signature (the set of interactions that the entity can engage in) and a behavioural protocol. In *Product Supply*, *Customer* denotes the specification of the interface *CR*, *Warehouse* that of *WR*, and so on.

Different formalisms are used in SRML for defining the behavioural protocols of each node depending on the nature of that node. It is out of the scope of this paper to present these specification formalisms, which can be found in [9]. For our purposes, it is sufficient to consider that each of the languages relies on the same computational model, which is based on the notion of *interaction event* discussed below.

SRML supports a number of interaction types to reflect both the interactions occurring within an enterprise and the typical business conversations that arise in SOC [4]. More specifically, interactions can be synchronous (i.e., the party waits for the

co-party to reply), or asynchronous (i.e., the party does not block). Typically, synchronous interactions occur in communications with persistent components, namely through uses-interfaces. We distinguish the following types of interactions, each described from the point of view of the party in which it is declared:

- ***snd*** and ***rcv*** are one-way asynchronous (send or receive) interactions.
- ***s&r*** and ***r&s*** are conversational asynchronous interactions (***s&r*** after "send-and-receive" is initiated by the party, and ***r&s*** is initiated by the co-party).
- ***ask*** and ***tll*** (resp. ***rpl*** and ***prf***) are synchronous interactions to obtain data (resp. to ask to perform an operation).

One-way interactions (***snd*** and ***rcv***) have an associated initiation event denoted by *interaction*🔔. Conversational interactions (***s&r*** and ***r&s***) involve a number of events exchanged between the two parties: once the initiation event *interaction*🔔 is issued by a party, a reply-event *interaction*✉ is sent by the co-party offering a deal; the party may then either commit to the deal (issuing *interaction*✓) or terminate the conversation (issuing *interaction*×); after committing, the party can still revoke the deal (issuing *interaction*✞), triggering a compensation. In this paper, we abstract from the parameters exchanged in the interactions, as they are not relevant for the analysis.

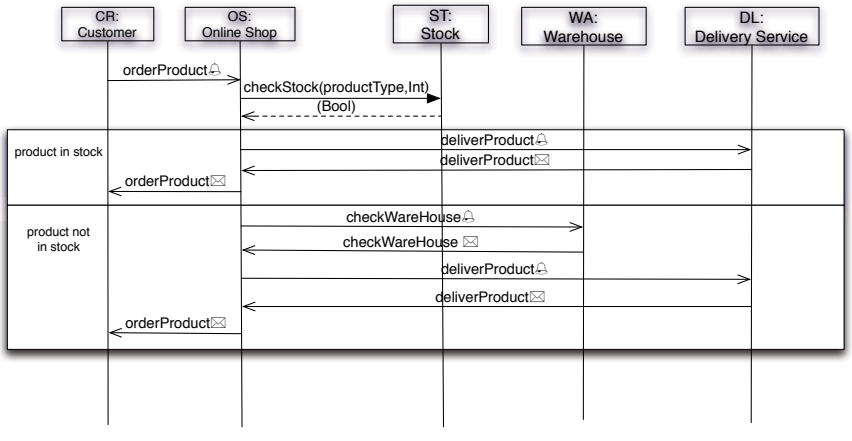


Fig. 2. Overview of the behaviour of module *Product Supply*

The behaviour of *Product Supply* is illustrated in Figure 2. The interaction *orderProduct* allows *CR* to ask the service to supply a product and receive a confirmation; the orchestrator *OS* checks for the product in the local stock with a synchronous interaction *checkStock*; if the product is not available in the local stock, *OS* requests the product to the warehouse service using the conversational interaction *checkWarehouse*. The interaction *deliverproduct* arranges for the product to be dispatched. In this scenario, *OS* may find the product in the stock (case *product in stock*); otherwise, it will contact the warehouse service *WR*. In any case the service will arrange the delivery of the product.

2.2 Modelling SLA Properties

SRML adopts a model for constraints on non-functional requirements in dynamically changing configurations based on c-semirings [5]. The constraints are used for modelling the SLAs involved in service discovery and selection. Our example uses a fuzzy c-semiring where the degree of satisfaction lies in the interval [0,1]. An external configuration policy *cs(M)* consists of: (1) a set of SLA variables $var_{SLA}(M)$, and (2) a set of constraints *SLA(M)*. The variables and constraints determine the quality profile to which the provided service and each discovered service need to adhere; they can represent QoS parameters such as cost, response time, delays, *inter alia*. Below we show *cs(Product Supply)*:

EXTERNAL POLICY

SLA VARIABLES

CR.COST, WR.COST, DL.COST, CR.LOCATION, WR.LOCATION

CONSTRAINTS

C_1: {CR.COST, WR.COST, DL.COST}

$Def_1(d,e,f) = \begin{cases} 1 \text{ if } d = e + f + 1.6 \text{ and } d \leq 6 \\ 0 \text{ otherwise} \end{cases}$

C_2: {CR.LOCATION, WR.LOCATION}

$Def_3(d,e)$= 1 if **distance**(d,e)<20, 20/**distance**(d,e) otherwise

The set var_{SLA}*(Product Supply)* includes three SLA variables – *CR.Cost*, *WR.Cost*, and *DL.Cost* – representing the transaction costs associated with the customer, the warehouse and the delivery service, respectively. Variables *CR.Location* and *WR.Location* represent the location of the customer and the warehouse. respectively.

The set *SLA(Product Supply)* includes two constraints – C_1 and C_2. C_1 expresses that the cost of the transaction for the customer should not be more than \$6 and that it consists of the sum of the cost of the warehouse transaction, the cost of the delivery service and a local cost of \$1.6. C_2 minimizes the distance between the customer and the warehouse – *distance(d,e)* is a function returning the distance between two locations. If the distance is less than 20 miles, the satisfaction is 1, otherwise the satisfaction is inversely proportional to the distance.

The aim of the quantitative analysis illustrated in Section 4 is to guarantee that C_1 holds up to a certain probability, for example, the cost of the overall service, defined as the business process occurring between the events *orderProduct*📤 and *orderProduct*📨, is less than or equal to 6\$ in 85% of the cases.

3 Modelling Cost in SRML

In this section, we extend SRML by annotating the different elements of a module with *cost rates*. A cost is considered to be the amount of money that a user will incur in when using a certain node or edge. Cost is modelled as a rate with exponential distribution – other distributions would require knowledge of higher moments and other parameters, which would require careful measurement or estimation; the exponential distribution requires only a single parameter – the average response time – which is more likely to be known.

The rate represents the coefficient r of the exponential distribution, which defines the probability for the cost to be lower than a certain value: $F_{Delay(a,b)}(t)=1-e^{-rt}$. Figure 3 shows the exponential distributions for the rates 3.0, 4.0, and 5.0. This diagram was obtained using the PEPA toolset. If, for example, the rate is 3, then the cost will be lower than \$2 in 50% of the cases and will be lower than \$5 in 100% of the cases. Naturally, a higher rate characterises a lower cost.

The Cost Policy for a module *M* includes the following *cost rates*:

- For every provides-interface *n∈provides(M): ProvidesRate(n).*
- For every requires-interface *n∈requires(M): RequiresRate(n).*
- For every *w∈edges(M): WireRate(w).*
- For every *n∈components(M): ComponentRate(n).*
- For every *n∈uses(M): UsesRate(n).*

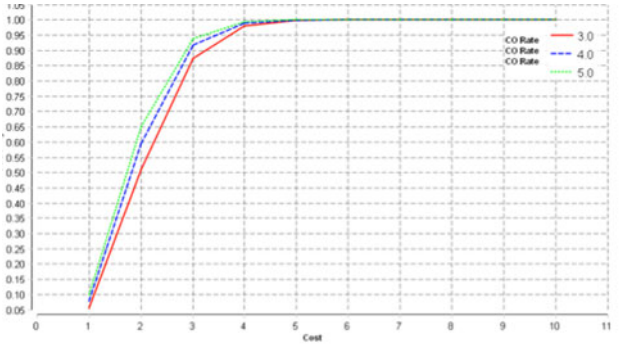


Fig. 3. Relationship between Cost and Rate

WireRate represents the cost a requester would incur to when using the wire. Similarly the cost for using a component-interface, requires-interface, Uses-interface is regarded as *ComponentRate*, *RequiresRate*, and *UsesRate*, respectively.

In order to make analysis more accurate, one may annotate each alternative branch of the sequence diagram representing the overall behaviour of *Product Supply* with the probability of engaging in that branch (where the sum of the probabilities for all branches of the same branching point is equal to 1). For example, we could annotate the sequence diagram of *Product Supply* in Figure 2 to model that the probability that the product is in stock is 0.6.

Once the nodes and wires of a module have been annotated, the aim is to derive, say, the cost of the business process between two events occurring in the provides-interface *CR* (e.g., *orderProduct*🖃 and *orderProduct*🖂) by considering the cascade of costs incurred between those two events. Figure 5 gives the overall cost of the activity between *orderProduct*🖃 and *orderProduct*🖂 (assuming that the product is not in stock), which is denoted with (1) and it is built upon the costs of wire *CO* – (2), component *OS* – (3), wire *OW* – (4), requires interface *WR* – (5), wire *OD* – (6) and requires interface *DL* – (7).

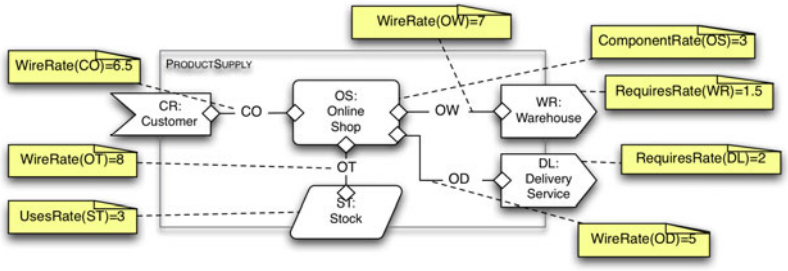


Fig. 4. SRML module annotated with *cost rates*

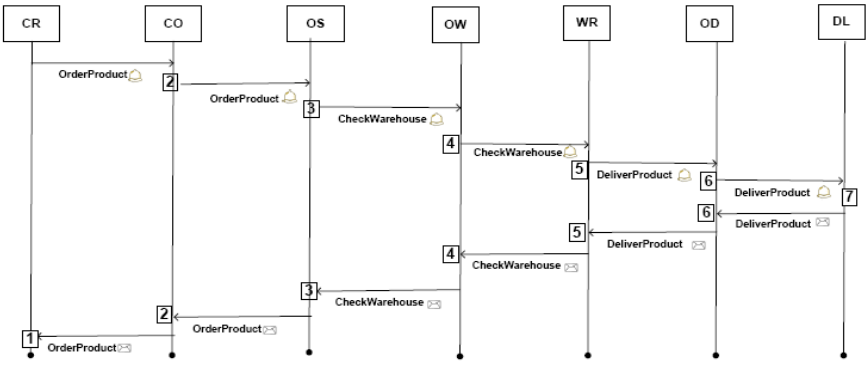


Fig. 5. Cascade of Costs in service *Product Supply*

4 Encoding SRML Modules into PEPA Configurations

SRML modules annotated with cost rates can be encoded in PEPA using a simplification of the encoding presented in [6]. A system is described in PEPA as a configuration involving a number of processes collaborating along a number of channels. Processes perform activities of the form $(\alpha,r).P$ where α is an action associated with rate r and P is the continuation. The encoding of *Product Supply* is

$$(CR||OS||WR||ST||DL)\langle L\rangle(CO||OT||OW||OD)$$

where each process on the left-hand side (e.g., *CR*) corresponds to a node of *Product Supply,* and each process on the right-hand side (e.g., *CO*) corresponds to an edge of *Product Supply*. *L* is called a *coordination se*t and includes the actions corresponding to each event triggered by any node. The encoding is modular in the sense that the process corresponding to each node/edge can then be obtained independently of the encoding of the other elements of the module (for example, the SRML component *CR* is encoded as a PEPA process *CR*, etc.).

The simplified encoding has been developed in [2]. In the case of components/requires-interfaces, where delays affect each interaction with the component/requires-interface, it considers a forfeit cost for using a component or a service and is more concise. We are currently working on the automation of the approach by extending the SRML editor to allow the annotations with rates, and by automating the encoding of annotated SRML models into PEPA models, creating a bridge between the SRML editor and the PEPA toolset.

5 Quantitative Analysis of Cost Properties

We perform quantitative analysis in three steps: (1) We derive the state space of the PEPA encoding to find any deadlock or unreachable state; (2) We perform passage-cost analysis to derive the probability of the cost of the business process inter-occurring between two activities; (3) We perform sensitivity analysis, which allows us to observe how changing the value of rates changes the cost.

As to (1), state-space derivation is necessary to find deadlock states in the system because the deadlocks prevent the analysis to proceed further. Passage-cost analysis cannot be performed until all the deadlock states are removed from the system. State-space analysis of *Product Supply* service modelled 11 elements (i.e., wires and nodes) and 20 deadlock free states resulted from the modelled interaction pattern. The approach scales to larger systems as the state space, in most cases, grows linearly with respect to the number of architectural elements. In fact, the flow of events of a SRML business process typically involves one or two architectural elements a time (i.e., the cases of parallellism are fairly limited in relation to the global number of architectural elements). This relieves us from considering all possible interleavings of events. As to (2), we want to analyse the cost of the business process between the occurrence of the event *orderProduct*🔔 (i.e., the order from the customer) and *orderProduct*✉ (i.e., the system's reply). The aim is to determine whether the system is able to guarantee that the cost, in 85% of the service invocations, is not greater than 6$. As to (3), we use feedback from step (2) to determine that where it best to invest effort in making one of the activities more economic. By changing the values of the rates, we can identify those nodes and wires that cause maximum change in cost.

Passage-Cost Analysis — The feature passage-time analysis provided by the PEPA Eclipse Plug-in is typically used to analyse the performance of a model when the rates represent the delay of each single activity. We use such functionality here to perform "passage cost analysis" by interpreting each rate in the model as a *cost rate* instead of a time rate. Passage-cost analysis produces a graph representing the cumulative distribution function (CDF) that plots the probability of having completed the passage of interest (e.g., *orderProduct*🔔, *orderProduct*✉) by a given cost bound. The CDF graph is illustrated in Figure 6. The black dot in the graph underlines the probability of a maximum cost of $6. The probability is less than 85%; in other words, in 85% of the cases we can only guarantee a maximum cost of $6.

Feedback and Reengineering — Sensitivity analysis was performed by changing each cost rate to a fixed number of possible values to see if any improvement could be

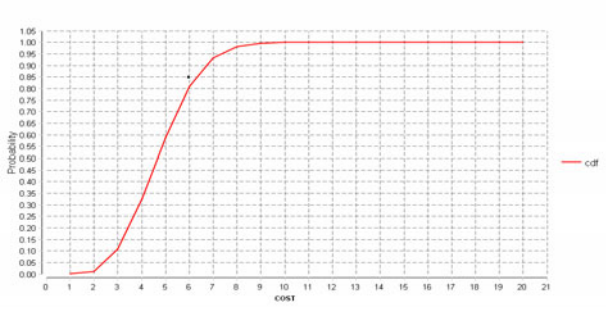


Fig. 6. CDF with initial Rates

found and the advertised SLA be satisfied. The cost rates affiliated with each node and wires are changed. There are four wires involved in the scenario *CO*, *OT*, *OW* and *OD*. The original cost rate of all wires is 3. We have changed the cost rates of every wire to two values below the true value (2,2.5) and two values above the true value (3.5,4). We noticed that all the wires brought the same change and were able to satisfy SLA at cost rate of 4. All the wires reduced the cost of transaction from 6.4$ to 6$. The nodes of the scenario are component-interfaces *OS*, ST, *WR* and *DL*. We have changed the cost rates of each of these nodes and tried to find the critical node. The original cost rate of all nodes is 4. We have changed the cost rates of all nodes two values below the true value (3,3.5) and two values above the true value (4.5,5). Only in the analysis of *OS* it was possible to satisfy the advertised SLA. Changes in the rates of *OS* brought the maximum performance change and reduced the cost from 6.4$ to 6$, as shown in Figure 7.

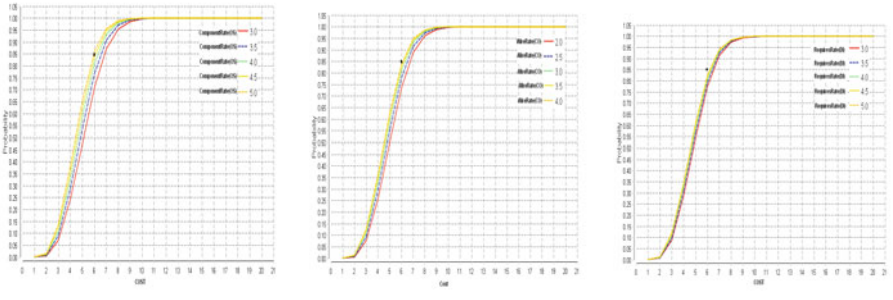


Fig. 7. Sensitivity Analysis of resp. *ComponentRate(OS), WireRate(CO)*, *RequiresRate(DL)*

6 Concluding Remarks

We have proposed an approach for analysing non-functional properties of service-oriented models as it arises in a number of business areas that, like logistics, are adopting service-oriented architectures. For this purpose, we have used the service modelling language SRML and the stochastic process algebra PEPA. We have

presented an encoding of a SRML model – *Product Supply* – of a typical logistics service into PEPA, and used the PEPA Eclipse Plug-in to perform quantitative analysis of cost-related properties, guaranteeing that SLAs can be met with a certain probability. We have also performed sensitivity analysis to find those critical components of the service that directly affect advertised SLAs. By finding these critical components, we can know where we should invest more to improve the performance of the system.

The proposed approach can be used for other non-functional properties, more specifically (following from our definition of rate) all non-functional properties that (1) can be represented by an exponential distribution and (2) affect the model as described by the annotation we have provided in Section 3. For example, we can use the proposed method to analyse *throughput*. In this case, *WireRate* would characterise the throughput of a wire (instead of the cost) where the rate represents the coefficient r of the exponential distribution of the form $F_{Delay(a,b)}(t)=1-e^{-rt}$ defining the probability that the throughput is higher than a certain value. In this case, a lower value of rate characterises a preferable setting, therefore a higher throughput. At the moment, the analysis of different properties should be performed independently. On the other hand, one could expect that the cost of a service could be directly proportional to other properties such as throughput. A topic for future development is supporting quantitative analysis of different inter-related properties.

Acknowledgements

We would like to thank Stephen Gilmore, Reiko Heckel for many useful discussions.

References

1. Alves, Maciel, Massa: Evaluating Supply Chains with Stochastic Models similar with Petri Nets. In: SOLI, pp. 1–6. IEEE, Los Alamitos (2007)
2. Azmat: Quantitative Analysis of Service Oriented Models. MSc Project Dissertation, Department of Computer Science, University of Leicester, UK (2010)
3. Balsamo, Di Marco, Inverardi, Simeoni: Model-based performance prediction in software development: a survey. IEEE Trans. on Software Engineering 30(5), 295–310 (2004)
4. Benatallah, Casati, Toumani: Web services conversation modeling: A cornerstone for e-business automation. IEEE Internet Computing 8(1), 46–54 (2004)
5. Bistarelli, Montanari, Rossi: Semiring-based constraint satisfaction and optimization. J. ACM 44(2), 201–236 (1997)
6. Bocchi, Fiadeiro, Gilmore, Abreu, Solanki, Vankayala: A Formal Approach to Modelling Time Properties of Service-Oriented Systems. In: Handbook of Research on Non-Functional Properties for Service-oriented Systems: Future Directions. IGI (to appear in Spring 2011), `http://www.cs.le.ac.uk/srml`
7. van Dam, Adhitya, Srinivasan, Lukszo: Critical evaluation of paradigms for modelling integrated supply chains. In: Proc. ESCAPE. Computers & Chemical Engineering, vol. 33(10), pp. 1711–1726. Elsevier, Amsterdam (2009)
8. Feng, Hong-yan: Evaluation of Logistics Service Provider Using Analytic Network Proces. ICNC 2, 227–231 (2007)

9. Fiadeiro, Lopes, Bocchi, Abreu: The Sensoria reference modelling language. In: Rigorous Software Engineering for Service-Oriented Systems. LNCS. Springer, Heidelberg (2010), http://www.cs.le.ac.uk/srml
10. Franceschini, Rafele: Quality evaluation in logistic services. International Journal of Agile Management Systems 2(1), 49–54 (2000)
11. Ghiani, Laporte, Musmanno: Introducing Logistic Systems. In: Introduction to Logistics Systems Planning and Control, ch. 1. Wiley, Chichester (2004)
12. Hillston: A Compositional Approach to Performance Modelling. Cambridge University Press, Cambridge (1996)
13. Iacob, Jonkers: Quantitative Analysis of Enterprise Architectures. In: Interoperability of Enterprise Software and Applications, pp. 239–252. Springer, Heidelberg (2006)
14. Iacob, Jonkers: Quantitative Analysis of Service Oriented Architectures. International Journal of Enterprise Information Systems 3(1) (2007)
15. IBM Developer Works. Understanding quality of service for Web services, http://www.ibm.com/developerworks/library/ws-quality.html
16. Kalepu, Krishnaswamy, Loke: Reputation = f(user ranking, compliance, verity). In: Proc. ICWS, p. 200. IEEE, Los Alamitos (2004)
17. Legeza: Measurement of logistics-quality. Per. Pol. Trans. Eng. 31(1-2), 89–95 (2003)
18. Liu, Ngu, Zheng: QoS computation and policing in dynamic Web service selection. In: Proc. WWW Conference on Alternate Track Papers & Posters, ACM, New York (2004)
19. Ran: A model for Web services discovery with QoS. SIGecom Exch. 4(1), 1–10 (2003)
20. The Open Service Oriented Architecture collaboration. Whitepapers and specifications, http://www.osoa.org (see also http://oasis-opencsa.org/sca)
21. Tian, Gramm, Naumowicz, Ritter, Schiller: A concept for QoS integration in Web services. In: Proc. WISEW, pp. 149–155. IEEE Computer Society, Los Alamitos (2003)
22. Tribastone: The PEPA Plug-in Project. In: Quantitative Evaluation of SysTems, pp. 53–54. IEEE, Los Alamitos (2007)
23. Xu, Cao: Logistics Service Quality Analysis Based on Gray Correlation Method. International Journal of Business and Management 3(1) (2008)
24. Yang, Wu, Li: The Empirical Research on Cluster Supply Chain Flexibility and Logistics Capacity. In: ICAL, pp. 1066–1071. IEEE, Los Alamitos (2009)
25. Vu, Hauswirth, Porto, Aberer: A Search Engine for QoS-enabled Discovery of Semantic Web Services. International Journal of Business Process Integration and Management 4(4), 244–255 (2006)

Event-Driven Services: Integrating Production, Logistics and Transportation

A. Buchmann[1], H.-Chr. Pfohl[2], S. Appel[1,*], T. Freudenreich[1,*], S. Frischbier[1,*], I. Petrov[1], and C. Zuber[2]

[1] Databases and Distributed Systems Group, TU Darmstadt
lastname@dvs.tu-darmstadt.de
[2] Chair of Management & Logistics, TU Darmstadt
pfohl@bwl.tu-darmstadt.de

Abstract. Today's production processes are characterized by global supply chains, short lifecycles, and an increasing personalization of goods. To satisfy the demands for agility we must integrate the production with the logistics processes and knowledge about the underlying transportation services and infrastructure. This requires continuous monitoring and reacting to events. Service-oriented architectures have provided a platform for structuring services within and across enterprises. However, for an effective monitoring and timely reaction to emerging situations it is crucial to integrate event processing and service orientation. In this position paper we show how event processing and service orientation can be combined into an effective delivery platform for an integrated coordination of the flow of goods. We show how simple events, e.g. RFID tag detections or simple sensor readings, can be integrated into abstract events that are meaningful to invoke logistics services and improve the celerity of responses. We propose filtering, aggregating, and on-the-fly analysis of the continuous flow of events and make events persistent in an event warehouse for auditability and input to future planning processes.

Keywords: Events, complex event processing, monitoring, services, event warehouse, agility, production, logistics, transportation.

1 Introduction and Status Quo

Integrating Production, Logistics and Transportation: Current trends in manufacturing show shortened product lifecycles and an increased personalization of goods with the resulting smaller lot sizes and more frequent retooling. New concepts like just in sequence production or the possibility of global sourcing and distribution have increased the complexity of logistics and the integration of transport systems [8]. In addition the recent consciousness about new objectives in the value creation – like the

* Funding by German Federal Ministry of Education and Research under research grants ADiWa (01IA08006) and Software Cluster EMERGENT, by Hessen Ministry of Higher Education, Research and the Arts under research grant LOEWE Dynamo PLV, and by Deutsche Post AG.

E.M. Maximilien et al. (Eds): ICSOC 2010 Workshops, LNCS 6568, pp. 237–241, 2011.

product's carbon footprint – have led to an extension of the cost-based optimizations of production and distribution strategies by ecological and social aspects. As a result, we must consider the problems of production, logistics and the most efficient use of the underlying transportation services and infrastructure in a comprehensive manner that allows us to optimize for multiple goals. Decentralized decision making and the flow of information across multiple domains and enterprise boundaries play a crucial role in this situation. Tomorrow's software systems have to be agile in terms of integration and adaption to meet the requirements of modern supply chain management.

Service-oriented Architecture (SOA) - Drawbacks and Opportunities: On this way towards dynamic processes, service orientation has become a key concept to enable modularization of information systems and the integration of legacy systems. By encapsulating a given functionality as a service and providing a standardized interface it becomes much easier to build new systems and to adapt to rapidly changing demands. However, current enterprise software systems based on SOA are mostly custom-tailored to the vital needs of a particular organization, reflecting individual workflows, semantics and contexts. Although SOA is a promising approach to reduce functional redundancy and syntactical complexity [4], inter-organizational integration is still hard to achieve. Furthermore, service orientation was conceived with a classical request-reply invocation paradigm in mind: the consumer of a service requests a service from a service provider and receives a response. This is the paradigm used in client-server architectures, such as typical database systems. In the case of services the invocation of a service can be either direct or through some form of mediator that helps in locating the proper service, but the initiation of an interaction is still an explicit request.

Introducing the Event-based paradigm: Pure service orientation with a request-reply interaction is ill-suited for today's cyberphysical systems in which streams of heterogeneous events from different domains and their associated data are continuously produced by sensors and embedded systems and can provide real time information on many operational entities. Typical low-level events are the RFID tag readings detected at a warehouse gate, the GPS coordinates of a truck, or the readings of a temperature sensor in a container with food or pharmaceutical goods. These simple, low-level events can be combined and more abstract and application relevant events can be derived from them. For example, knowing the GPS position of a truck and the traffic conditions ahead, we can derive a complex event that tells us that the truck will arrive two hours late at its destination. The responses to such detected situations can and should be packaged as services. However, the manner in which they are *automatically* invoked is different. Events are produced and many different parties within different contexts may show interest in certain events. They will be notified that an event happened rather than having to ask for it. This reduces the latency between the occurrence of an event and the proper reaction. It also avoids unnecessary polling cycles.

Integrating SOA with Events: These event-driven services combine the benefits of complex event processing (CEP) and SOA [7]. The resulting event-driven architecture (EDA) [1] encompasses both approaches and tries to provide a common framework to cover requirements both from inside an organization as well as from the outside. Companies are starting to recognize the benefits of this approach. However, there is still a huge gap between potential and realization. Software providers are trying to

develop the necessary event processing platforms. These are needed, just as database management systems were needed as common platforms decades ago. The end users are trying to understand how to use these platforms to their advantage; at this many open issues exist on both sides [2].

Technological Requirements: From a technological point of view there is the need for powerful notification services that can be distributed and scale appropriately [3], well-defined and standardized languages for event definition, operations on event streams, efficient persistence mechanisms for massive streams, integration of transactional behavior in stream processing, heterogeneity, and security and privacy are just some issues.

Application Requirements: From an application point of view the issues are, among others, the definition of relevant domain-specific events, mechanisms for combining the flood of events with operational data kept in databases, persisting events for auditability purposes and for future planning, decide on an appropriate level of aggregation for events, and how to design and deploy in an effective manner software systems that take full advantage of event-driven services.

Related Projects: In this position paper we sketch an approach that we are investigating in three joint projects between university researchers, major software companies, and big players in the logistics, transportation and production domain. In the Dynamo PLV project we address the issues how to integrate the three domains, production, logistics and transportation, and how to provide the proper infrastructure both from an IT perspective as well as domain-specific. In the ADiWa project we concentrate on issues of quality of service in event delivery and composition. In the Software Cluster project EMERGENT we concentrate on the lifecycle of events and software engineering of event-driven systems to ensure a high level of interoperability between systems based on different domains and platforms.

2 Events

The logistics domain has pioneered many of the automated tracking and tracing applications to afford a more efficient supply chain management. Some sectors, such as air cargo, have successfully defined and implemented a set of domain-specific events and milestones in the context of the Cargo 2000 program to simplify quality management [6]. Examples of such milestones are “Pick Up from Customer” or “Departure” (airline event). Existing software can monitor standardized messages and raise failure alerts or produce route maps with comparisons of planned and actual milestones. This is a very good first step, but in its present form is only applicable to the air cargo sector. Our goal is to develop a platform that would allow us to extend the Cargo 2000 approach to other transportation sectors, such as rail, trucks and combined traffic & transport as well as the integration of intralogistics, and to provide the necessary event definition language.

Further, Cargo 2000 milestones represent logistics processes only with a coarse granularity. Thus another goal is to integrate low-level events that are automatically detected into the whole process. For example, GPS data or events from the German toll system that tracks all trucks above 12 tons when they use the German Autobahn

system. Besides the technical problems derived from such a use of truck sightings by the network of installed sensor nodes, it raises a series of interesting privacy and security issues that must be addressed.

The goal is to exploit the same events in many different applications. To this end, a flexible subscription mechanism is needed to enable new services to subscribe to already available events. Furthermore, different services may require existing events as input to compose more complex events in different constellations. We show this process schematically in Figure 1. Underlying this simplified schematic is the need for a well-defined event algebra, an event consumption policy, and an event lifecycle management. The event algebra defines the operators that allow us to combine events, the consumption policy determines in which order events must be consumed (e.g. chronologically or always the latest) and the lifecycle management determines, when an event can be discarded and what events must be made persistent. In [5] we define all the basic terms and identify the event processing requirements of many applications in different domains, including logistics, supply chain management and traffic monitoring and management. This work must be expanded to include event flows in production environments and shop floor control.

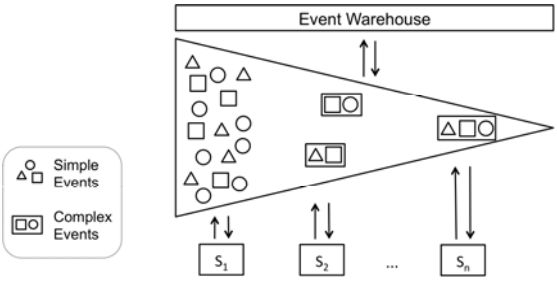


Fig. 1. Complex Event Processing by Services S_1 to S_n

3 Event Warehouse

While traditional data management is concerned with storing transactional data (OLTP) and data warehousing is concerned with consolidating operational data for analysis purposes (Business Intelligence), event processing deals with on-the fly processing of streams of heterogeneous events or rather their representation, the event objects. Events are filtered, aggregated and analyzed on the fly. The resulting business events must be reported in a timely manner.

Many events have only transient importance or do not have subscribers and can be discarded. It would unnecessarily clutter the database to store every RFID tag reading or very dense traces of truck trajectories. On the other hand, for planning and simulation purposes as well as auditability all the pertinent events must be recorded.

Finding the right granularity for making events persistent is one of the goals of our current research. Equally important is to determine what underlying events and what context information must be made persistent when storing higher-level business events that were derived and used in the decision making process.

A comprehensive approach to production, logistics and transportation requires the design and execution of complex simulation models. The weak point of many models is the unreliable and incomplete set of input data and the assumptions that are made to compensate for it as well as the missing interfaces of each model to communicate with each other. Therefore, the idea of event warehouses (EW) is a new trend in research [9, 10]. The EW plays a central role in providing a complete and realistic set of events and traces that can be used to execute what-if scenarios, to calibrate the models that are being used by comparing their predictions against recorded data, and in the development of new adaptive strategies.

4 Outlook

Event-driven services are the key to integrating SOA and CEP. Services that are triggered by events rather than invoked, be it manually or on a timed basis, offer a much better responsiveness to business needs. By making events into first class citizens, we can capture the dynamics of complex production, logistics and transportation systems. The event warehouse serves as a repository of meaningful business events as substructure of decisions with all its impacts. It thus captures the result of on-the-fly analytics as well as the underlying system dynamics and context information required for reconstruction of relevant situations. It provides the necessary inputs to models for comprehensive simulation of complex production, logistics and transportation systems.

References

1. Chandy, K., Schulte, W.: Event Processing: Designing IT Systems for Agile Companies. McGraw-Hill, Inc., New York (2010)
2. Frischbier, S., Sachs, K., Buchmann, A.: Evaluating RFID Infrastructures. In: 2nd Workshop RFID Intelligente Funketiketten - Chancen und Herausforderungen, Germany (2006)
3. Guerrero, P., et al.: Pushing Business Data Processing Towards the Periphery. In: 23rd IEEE International Conference on Data Engineering, Turkey (2007)
4. Herr, M., Bath, U., Koschel, A.: Implementation of a service oriented architecture at deutsche post MAIL. In: Zhang, L.-J., Jeckle, M. (eds.) ECOWS 2004. LNCS, vol. 3250, pp. 227–238. Springer, Heidelberg (2004)
5. Hinze, A., Sachs, K., Buchmann, A.: Event-based applications and enabling technologies. In: 3rd ACM International Conference on Distributed Event-Based Systems, USA (2009)
6. IATA - International Air Transport Association. Cargo 2000 (2010), http://www.iata.org/whatwedo/cargo/cargo2000/Pages/index.aspx
7. McGovern, J., et al.: Enterprise Service Oriented Architectures: Concepts, Challenges, Recommendations. Springer-Verlag New York, Inc., New York (2006)
8. Pfohl, H.-C., Buse, H.P.: Inter-organizational logistics systems in flexible production networks: An organizational capabilities perspective. International Journal of Physical Distribution & Logistics Management 30(5), 388–408 (2000)
9. Roth, H., et al.: Event data warehousing for Complex Event Processing. In: 4th International Conference on Research Challenges in Information Science (2010)
10. Winter, R., Kostamaa, P.: Large scale data warehousing: Trends and observations. In: 26th IEEE International Conference on Data Engineering, USA (2010)

Preselection of Electronic Services by Given Business Services Based on Semantic Concept Correspondence Applied for the Logistics Domain

Rolf Kluge[1,2]

[1] Information Systems Institute, University of Leipzig,
Grimmaische Str. 12, 04109 Germany
rkluge@wifa.uni-leipzig.de
[2] Department of Computing, Macquarie University,
Sydney, NSW 2109, Australia
rolf.kluge@mq.edu.au

Abstract. Service oriented environments consist of electronic and business services. Business services encapsulate core business activities whereas electronic services support the operation of business services by means of enterprise software applications. In environments with a large number of business and electronic services, methods for the selection of electronic services for certain business services are needed. This paper presents a conceptual approach for a preselection of electronic services that potentially fits the needs of a given business service based on semantic concept correspondence. The approach is based on the hypothesis that the higher the correspondence between semantically described concepts of business and electronic services the higher the probability of a match between them. This will support decision making during electronic service evaluation. The approach is elaborated based on certain requirements and evaluated by a use case scenario of the logistics domain.

Keywords: Service Evaluation, Service Discovery, Preselection, Semantic Web services, Ontology Matching, Logistics, Transportation Management.

1 Introduction

Companies are constantly faced with changing market conditions and threats, new competitive pressures in terms of cost, time, and flexibility, and ever changing regulations that require compliance. In order to cope with those conditions, companies outsource internal applications to external service providers in order to focus on the growth of their core activities and competencies. This holds particularly true in the area of logistics. Outsourcing includes business functions such as warehousing, transportation, transshipment, order management, etc., but also computing functions such as enterprise software applications. Furthermore, service providers e.g. third and fourth party logistics[1] engage subcontractors in order to fulfill certain functions. This

[1] "A fourth-party logistics provider is an independent, singularly accountable, non-asset based integrator of a clients supply and demand chains." [1].

E.M. Maximilien et al. (Eds): ICSOC 2010 Workshops, LNCS 6568, pp. 242–252, 2011.

leads to new markets and business models for providing business and computing functions as well.

With the advent of service orientation an important design paradigm that significantly eases outsourcing was established. Service orientation utilizes services as basic abstractions of business and computing functions. Services are defined as self-contained, loosely coupled entities that encapsulate a limited piece of functionality. Services are reusable, are able to be composed, and they provide a well defined external interface. Service orientation as a design paradigm for business and computing functions lets enterprises easily integrate externally provided services into internal processes and promises a number of benefits – among them flexible re-configuration, dynamic binding, easy access to heterogeneous resources and processes, transparency across implementation details and last but not least a relatively stable set of standards for aspects such as interface, orchestration, choreography or contracting.

Research on service orientation has been conducted in various directions. One of them is the application of the service oriented design paradigm towards business concerns and capabilities by means of business services (BS) [2] and towards encapsulating computing systems, information systems and software applications by means of electronic services (ES) [3]. An area which received insufficient support so far is the alignment of business and electronic services. Business and electronic services are naturally interrelated as ES provide the fundamental support by means of data, information and processing. However, in environments with a high number of BS and also usually a high number of ES that can be bound to support these BS the decision on which ES candidate provides the most suitable support for a certain BS is not trivial.

In order to provide decision support in the context of service evaluation, this paper presents a conceptual approach for a preselection of ES for given BS based on the semantic correspondence of concepts. The approach is based on the hypothesis that the higher the correspondence between semantically described concepts the higher the match between a business and an electronic service. The paper presents requirements, derives basic constituents and illustrates the approach by an example in the area of logistics.

The remaining of this paper is structured as follows: First of all, requirements as well as the approach and its constituents are described in chapter 2. Basically, the approach contains the three building block models, and preselection logic. These building blocks are examined in detail in chapter 3. Section 4 shows related work and chapter 5 draws a conclusion and mentions future work.

2 Requirements and Conceptual Approach

For preselection of ESs for given BSs based on their semantic concept correspondence a number of generic requirements must be satisfied. These requirements are described below and will be used as the basis for deriving the constituents of the approach. *Conceptualization & formalization*: Calculating of semantic concept correspondence of subjects such as business and electronic services requires that relevant aspects of them are represented explicitly or implicitly. *Comparison*: On the basis of formal conceptualizations, a general mean to compare both subjects is necessary. Thus, conceptualizations and their formalization of business and electronic services

need to be aligned to each other in a way that some matching logic can conclude about their semantic correspondence level. *Quantification*: Once conceptualizations of business and electronic services are aligned to each other by means of comparability, processing logic that is able to determine a semantic correspondence level between business and electronic services must be applied. *Aggregation*: Since the number of subject conceptualization entities may be high and lead to multi-dimensional vectors, aggregation logic that consolidates correspondence levels on a concept level towards service level correspondence is needed. *Visualization*: The results of determining correspondence levels for a large number of business and electronic service subjects on a service-level may still be complex and due to its numeric and multi-dimensional character difficult to interpret by humans. Thus, a visualization metaphor for representation of large scale correspondence levels and formatting them towards easy interpretation by humans for decision making is required. *(Semi-)Automation*: Since the selection and evaluation of electronic services for a certain business service can be fulfilled manually as well, the approach should facilitate automation in order to gain cost and time efficiency. However, since full automation may lead to error-prone results the approach should leave space for manual steps.

The constituents of the approach for the preselection of ESs for given BSs due their business semantics are derived from requirements described above. The conceptual approach is illustrated by Fig. 1.

For formalization, conceptualization and comparison reasons a service meta-model (called service comparison meta-model SCMM) is needed. The SCMM contains for example underlying processes, resources, capabilities, etc. Since there are several service models preexisting (cf. 0. Related Work), the elements of the SCMM can be derived from them. However, the SCMM must fulfill the following requirements: First, the meta-model has to be commonly applicable for BS and ES in order to satisfy basic comparability. Second, basic service elements, such as capabilities, processes, resources, etc., and aspects such as internal and external, static and dynamic characteristics have to be considered. Third, the SCMM must leave space for semantic variation.

The SCMM is applied to several BSs and ESs, which results in several Business Service Comparison Models (BSCM) x_i and Electronic Service Comparison Models (ESCM) y_j. Each BSCM x_i contains many BS Concepts $n_{i,k}$, whereas i is the index of the BS and k is the index of the concept. Analogously, each ESCM y_j contains many ES Concepts $m_{j,l}$, whereas j is the index of the ES and l is the index of the concept. These concepts have to be extracted, normalized and formally described for further processing.

Next, matching logic can be applied in order to conclude about the correspondence level for each $n_{i,k}$-$m_{j,l}$-combination. The application of the matching logic results in a correspondence value $c_{i,j,k,l}$. The correspondence value is a quantification of the similarity between a certain BS concept $n_{i,k}$ and a certain ES concept $m_{j,l}$. Since there are many (k*l) possible concept combinations for each BS-ES-combination (i*j) automation is needed. Automation is expressed by the function $f(n_{i,k}, m_{j,l})$. After having similarity measures on semantic concept level, the values have to be aggregated for a certain BS-ES combination in order to conclude about similarity on service-level and service-element-level. The aggregation is embodied by the function $f(c_{i,j,1,1}, c_{i,j,1,2}, \ldots, c_{i,j,k,l}, \ldots c_{i,j,K,L})$ taking into account each $c_{i,j,k,l}$ value. There is a quantifiable similarity value for each combination $s_{i,j}$ as a result of the aggregation.

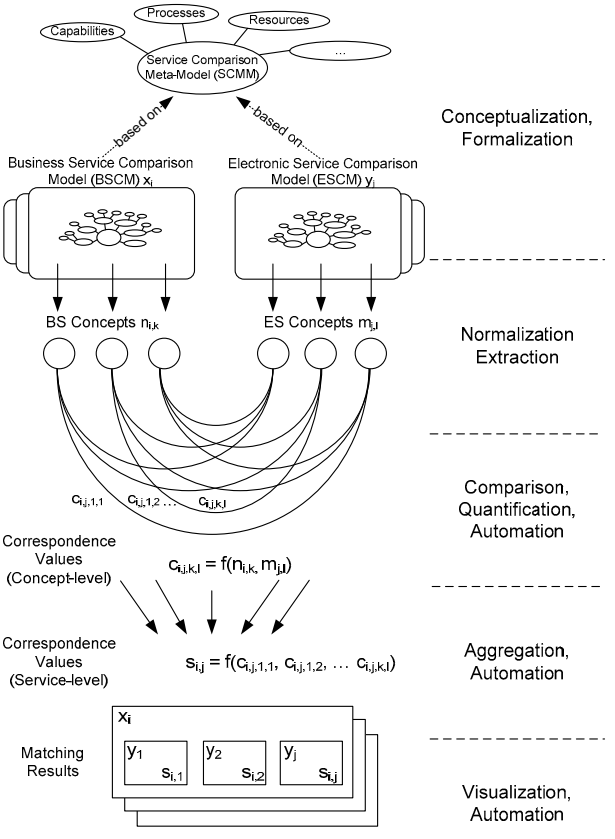


Fig. 1. Conceptual Approach for Preselection

For achieving the goal of supporting decision making the pure values are not sufficient. They have to be represented and visualized. This is depicted by the ranked list of high potential ESs for each BS in the lower part of Fig. 1.

3 Constituents in Detail

The conceptual approach the building blocks: SCMM as well as its instantiations (i.e. BSCM and ESCM), and preselection logic. These are examined in detail below. Visualization exceeds the scope of this paper.

3.1 Service Comparison Meta-model

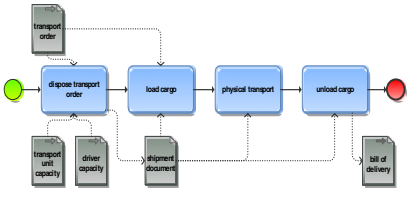
A meta-model describes the structure of a possible model including constructs, relationships between them as well as restrictions and modeling rules. Furthermore, a meta-model describes the structure in an abstract way. The concrete syntax of that is provided by the modeling-language. Whereas the abstract syntax (i.e. the meta-model) provides the foundation for the automated and tool-supported processing of a model,

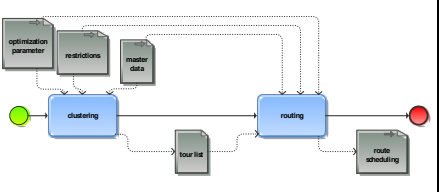
the concrete syntax provides concrete constructs in a textual or graphical manner that is used for modeling [4]. Adapting these definitions for the preselection approach, the meta-model defines common elements for comparison. As already mentioned, the meta-model is called service comparison meta-model (SCMM). Instantiations of the SCMM lead to several BS- and ES-models, which describe each service partly, but contain only comparable aspects. The BS-model is called Business Service Comparison Model (BSCM) and the ES-model is called Electronic Service Comparison Model (ESCM). For representation purposes a modeling-language will make BCSM and ECSM manageable. Since modeling-language as well as the BSCM and ESCM have to be in accordance to the SCMM and since such a SCMM does not exist so far, the elements of the SCMM has to be elaborated. There are several existing approaches for service modeling (cf. 0. Related Work). Thus, the SCMM can be derived from them in accordance to the three already mentioned SCMM requirements (cf. chapter 3). However, since the conceptual approach is widely independent from the underlying SCMM and the fact that a SCMM is hard to predefine, because there are always alternatives, we provide a very generic and simplistic meta-model for this document. The SCMM is derived from [5]. Karakostas and Zorgios define that a service contains capabilities, processes, and resources. A resource is either a tangible (e.g. a truck of a logistics company) or an intangible (e.g. transport order information). Resources are governed by the service provider and must be in place in order to fulfill a service. A process coordinates resources, is created and managed by the service provider, and issues capabilities. A capability embodies the functionality of the service (cf. [5]). These service elements supplemented by actions, which are atomic operations of a process, including their relationships, leads to the SCMM. For simplification purposes services, capabilities, actions and resources are characterized by their name and a description. The process, which comprises actions and resources, is described by a process description as a BPMN diagram [6]. The textual expression of name and description as well as the BPMN diagram and its graphical representation form the modeling-language of the SCMM.

An example for a BS is a standard transportation service fulfilled by a freight forwarding company. The result of the BSCM by using the SCMM is depicted on the left hand side of Table 1. The service is a truck cargo transport. The capability of the service is that the company fulfills all actions from transport planning to transport execution. In detail, the actions are plan transport order, load cargo, transport cargo physically, and unload cargo at destination. The process is depicted as a BPMN. The process has a start as well as an end event and contains all mentioned actions. Further, the process contains resources as inputs and outputs of actions. Resources are transport order, transport unit capacity, driver capacity, shipment document and the delivery bill. On the other hand, there is an example for an ES taken from the SAP® Transportation Management module “Planning and Optimization”. Information about SAP® Transportation Management is gained from the user interface of the software and from literature [7]. This results in the ESCM illustrated on the right hand side of Table 1. The ES is a tour planning service, which supports the planning of cargo transport tours. The service has the capability to plan and optimize tours, which means that certain transport orders are allocated to certain transport units and the optimal route, i.e. the optimal sequence of transport locations, is calculated. The service comprises two actions – clustering and routing. Clustering comprises the

transport order planning, the tour planning, and engaged vehicle and driver planning. The service process contains both actions and some information resources as input and output parameters. The resources are optimization parameters, restrictions, master data, tour list, and route scheduling. Each resource element has sub-resource elements as shown in the description.

Table 1. BSCM and ESCM example

Business Service Comparison Model	
Name	*Description*
1.1 Service	
truck cargo transport	transport of cargo via truck
1.2 Capability	
transport planning and execution	Planning and execution of transport
1.3 Action	
plan transport order	transport order-, route-, resource planning
load cargo	load truck at start location
physical transport	physical transport (start-dest)
unload cargo	unload truck at destination
1.4 Process	
	
1.5 Resources	
transport order and destination	contains number, weight and size of cargo, as well as start
transport unit capacity	number of available transportation units
driver capacity	number of available drivers
shipment document	contains the transport order document, dates
delivery bill	shipment document signed by the receiver

Electronic Service Comparison Model	
Name	*Description*
2.1 Service	
tour planning	planning tours for cargo transport
2.2 Capability	
tour planning optimization	automatic transport unit allocation for transport orders, calculation of optimal sequence of locations
2.3 Action	
clustering	transport order-, tour-, and resource planning
routing	calculate optimal sequence
2.4 Process	
	
2.5 Resources	
optimization parameter	comprises the optimization type (“min transportation units”, “min distance”, “min operation time”, …)
restrictions	comprises “capacity of transportation units”, “priority of transportation order”, “time slot for pickup”, and “time slot for delivery” …
master data	“transport network”, “resource information, “transport order” …
tour list	a list of tours; transport orders are assigned to a certain tour …
route scheduling	several routes; each route comprises an ordered set of transport activities; transport activities are …

3.2 Preselection Logic

Based on BSCM and ESCM descriptions, preselection logic can be applied in order to measure the terminological correspondence between BS and ES. Formalization, normalization and extraction of concepts are necessary in order to make BSCM and ESCM descriptions machine processable, first. Second, comparison logic will be applied, in order to measure correspondence for each single concept. Third,

aggregation logic will be applied in order to measure the correspondence on element level, i.e. elements that are defined in the SCMM.

The description of BS and ES depicted in Table 1 are semi-formal and cannot be processed automatically so far. Thus, the BS and ES description has to be transformed into a formal model. For conceptual comparison, it is suitable to use semantic languages, which provide the expression of concepts and their relationships. An appropriated semantic language is OWL DL [8]. The transformation from the service description to a semantic representation of concepts is a straight-forward procedure, since there are Ontologie Engineering methodologies [9] available. However, some special cases have to be handled. Thus, the following rules will be applied for formalization and concept extraction within this document: (1) single nouns are single concepts, (2) compound nouns are split into their parts and treated as single nouns, (3) verbs are normalized as nouns, (4) adjectives and copulas are skipped and (5) each concept appears only once. Taking the five rules into account and applying the METHONTOLOGY [9] results in the semantic formalization is shown below. Fig. 2 shows a representation of the ontology created with Protégé[2]. Regarding the BSCM, Fig. 2 shows the SCMM elements on the left side, including the service element, the capability element, the action element, the resource element and the process element (cf. ellipses with dotted line). Since a process contains resources and actions, the description of a process is the summation of the resource elements description and the action elements description. The description statements are next to the SCMM elements. These are connected as stated in the description itself. Furthermore, there are extracted concepts on the right hand side of the BSCM description statements (cf. ellipses with dashed lines). These concepts are extracted following the five rules mentioned above. Such a OWL DL description exists for the ESCM as well (cf. right hand side of Fig. 2). All in all there are 29 single concepts extracted from the BSCM and 52 concepts extracted from the ESCM.

After the extraction of single concepts, correspondence values between them are measured. Several matching algorithms and strategies for measuring similarity between concepts are presented in literature (cf. 0. Related Work). However, for simplification purposes we chose a basic matching algorithm for this example (similarity 1 if concept names are equal and similarity 0 if concept names are different). Out of 29 single business service concepts there are 16 matches to single electronic service concepts (e.g. cargo, transport, plan, order, location, etc.). Next, single results have to be mapped to their roots in order to reveal matching results on combined conceptual level, service-element level (i.e. capability, process, action, and process), and service level. Thus, single matching results have to be aggregated on a combined conceptual level first. As an example the service description “truck cargo transport” contains three single concepts. Two of them – cargo and transport – match to single electronic service concepts. As an aggregation the combined concept “truck cargo transport” matches to two third (2/3) to the electronic service. In a similar way the concept match of levels above is calculated. As an example, the resource “transport order” contains the concept cargo and cargo contains weight and size. Weight matches 0/1, size matches 0/1 as well, cargo matches 1/1, start location matches 2/2, destination location matches 2/2, and transport order itself matches 2/2. The match on transport

[2] http://protege.stanford.edu/

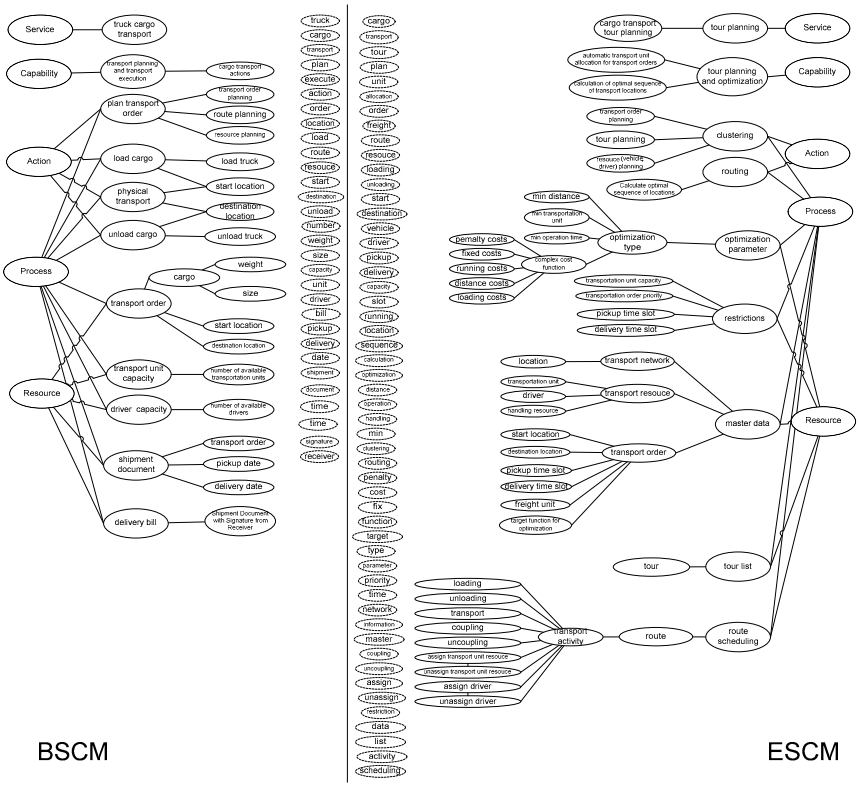


Fig. 2. BSCM and ESCM example as OWL

Table 2. Matching results on Service-Element Level

Element	Matching value
Service	0.6667
Capability	0.7143
Action	0.9200
Process	0.7222
Resource	0.5517

order level is 0.7778 (=0*1/9 + 0*1/9 + 1*1/9 + 1*2/9 + 1*2/9 + 1*2/9). In the same way a matching on a service-element level is calculated. Results are depicted in Table 2. The aggregated conceptual matching value on a service level is 0.7188 which means that the mentioned business and electronic service match with 71.88% based on their single concepts (by using this particular procedure). However, the value of 71.88% as a standalone statement leaves space for interpretation. The question whether the electronic service supports the business service or not is not answered. This can only be answered by applying the approach to several electronic services. Moreover, not only the end result of 71.88% has to be considered. The sub results on

service-element levels as well as certain concept matching results are interesting for a detailed evaluation. In order to support interpretation on various levels visualization is needed, though this is not part of this paper.

4 Related Work

Electronic Service evaluation and discovery is an important topic in SOA. The SOA basic model addresses this by an own entity called service registry (also known as service broker). A service consumer can discover the registry in order to find a service that fits its concerns (cf. [10]). Basic mechanisms, such as UDDI [11], support the search for Electronic Services based on keywords only. However, there are more sophisticated approaches based on semantic Web service (i.e. ES) description, such as WSMO [12] or OWL-S [13]. Semantic Web service discovery is much more precise, since it is based on the definition of Goals. These Goals have to be defined explicitly in terms of functional and non-functional parameters. However, there are divergent opinions about goals as well, since "service requesters are not expected to have the required background to formalize their goals" [14]. This is true due to the fact that business people (i.e. potential service requestors) know their business, but usually do not know what to expect from a Web service. Further, goals are defined by preconditions, assumptions, postconditions and effects (cf. [12]) using logical languages, but business people are not familiar with logical rules. Moreover, one might think that BSs are similar to Goals, but they are not. Goals define expectations on a (sought-after) Web service explicitly. In contrast, BSs describe business cases, which might implicitly contain requirements on a ES. With this, the approach for preselection of ES by given BS based on semantic concept correspondence fits in the gap between a keyword based service discovery and service discovery mechanisms based on semantic Web services. The approach is more precise than keyword based discovery, since not only keywords but the terminology of the business in terms of BSs as well as semantic concept correspondence are taken into account. In contrast it is less precise than a sophisticated semantic Web service discovery. However, the approach is reasonable for preselection purposes, i.e. reducing a large number of available ESs to a significant smaller set, which makes evaluation less complex and less expensive. Furthermore, it does not rely on Goals which might be hard to define and which are described in a language business people do not understand. This is similar to [15], where business process models are used for ES discovery. However, [15] relies on a common domain ontology, which is not considered here. Furthermore [15] refers to [16] for calculating matching degrees, but does not provide any matching or aggregation algorithm. More related work can be found within the areas of the three building blocks service comparison meta-model, preselection logic and visualization. For instance, there is related work in service modeling focused on business concerns (e.g. [17]). Most of these have their root in the marketing area. Further, there are service models and frameworks for ES (e.g. [13]) and there are even approaches that promise to be independent from business and technical concerns (e.g. [18]). Concerning preselection logic there are accepted workings for semantic concept extraction (e.g. [9]). Furthermore, there are algorithms for ontology matching as well (cf. [19]). This area provides techniques for quantifying similarity between semantic concepts. There are several matching techniques, which can be divided into element- and structure-level

techniques at first stage and into syntactic, external and semantic techniques at a second stage. However, single matching techniques are not sufficient. Usually, matching techniques are combined in order to get a proper matching result. There are sequential matching strategies, which put some matching techniques and variations into a sequence, as well as parallel matching strategies, which perform matching techniques at the same time and aggregate results afterwards (cf. [19]). Last not least, there are approaches for visualizing multi-dimensional results, such as heatmap visualization [20].

5 Conclusions and Future Work

Business services describe business concerns structured as services according to the service-orientation paradigm. Electronic services describe technical aspects as services (i.e. according to the service-orientation paradigm) with a special focus on enterprise software applications. Electronic services support operations of business services. Assuming there is a large number of business and electronic services, the evaluation of an electronic service that supports a business service best is time and cost consuming. This paper provides an approach which supports the evaluation process. It is based on the hypothesis that a certain electronic service matches a certain business service if electronic and business services are similar in their semantic concepts. The approach provides a procedure for measuring similarity between business and electronic services on conceptual level.

The application of the approach to a certain business service and a certain electronic service shows that there is a variety of possibilities for conceptualization, matching logic and aggregation logic. The example use case from the logistics domain resulted in a 71.88% conceptual match between a certain (logistics) business and a certain (logistics) electronic service. It is obvious that this value can vary using different aggregation functions, matching algorithms or even different descriptions of the same use case. Thus, it cannot be attested that a higher matching results represents a better combination than a lower matching results in each case, especially when values are very close to each other (e.g. 75% and 73%). This is due to the fact that description might be vary or imprecise. However, assuming that all BS and ES description are truly been made and complete, it can be attested that a BS-ES-combination with a higher matching value has higher probability to be a better solution than a BS-ES-combination with a significant lower matching value. Further, with the same settings and the same business service the approach reveals its significance by its application to a large number of electronic services.

Future work will focus the application of the approach by varying service meta-models, conceptualization methods, preselection logic (in terms of matching algorithms, matching strategies and aggregation logic functions), and visualization possibilities. Moreover, since the tool does not exist so far, this will be developed for handling, application, analyzing, as well as result interpretation.

Acknowledgment

The work presented in this paper was partly funded by the German Federal Ministry of Education and Research under the projects InterLogGrid (BMBF 01IG09010F) and Logistics Service Bus (BMBF 03IP504).

References

1. Win, A.: The value a 4PL provider can contribute to an organisation. International Journal of Physical Distribution & Logistics Management 38(9), 674–684 (2008)
2. Hering, T., Ludwig, A., Franczyk, B.: Proliferation in Service Types: Towards a Unifying Taxonomy of Service Types. In: International Symposium on Services Science, Leipzig (2009)
3. Papazoglou, M.P., Georgakopoulos, D.: Service-oriented computing - Introduction. Communications of the ACM 46(10), 24–28 (2003)
4. Petrasch, R., Meimberg, O.: Model-Driven Architecture: Eine praxisorientierte Einführung in die MDA, vol. 1. Dpunkt Verlag (2006)
5. Karakostas, B., Zorgios, Y.: Engineering Service Oriented Systems: A Model Driven Approach. Idea Group Publishing, USA (2008)
6. Miers, D., White, S.A.: Bpmn Modeling and Reference Guide. Future Strategies Inc. (2008)
7. Lauterbach, B., et al.: Transportation Management with SAP®TM, vol. 1. Galileo Press, Bonn (2009)
8. Lacy, L.W.: OWL: Representing Information Using the Web Ontology Language. Trafford Publishing, Crewe (2005)
9. Gómez-Pérez, A., Fernández-López, M., Corcho, O.: Ontological Engineering. In: Advanced Information and Knowledge Processing, vol. 1. Springer, Berlin (2004)
10. Papazoglou, M.P.: Web Services: Principles and Technology. Prentice Hall, Essex (2007)
11. Clement, L., et al.: UDDI Version 3.0.2
12. Roman, D., Lausen, H., Keller, U.: Web Service Modeling Ontology - WSMO Final Draft (D2v1.3). DERI, Innsbruck (2006)
13. Martin, D., et al.: OWL-S: Semantic Markup for Web Services, W3C Member Submission
14. Keller, U., et al.: Automatic Location of Services. In: Gómez-Pérez, A., Euzenat, J. (eds.) ESWC 2005. LNCS, vol. 3532, pp. 1–16. Springer, Heidelberg (2005)
15. Markovic, I., Karrenbrock, M.: Semantic web service discovery for business process models. In: Weske, M., Hacid, M.-S., Godart, C. (eds.) WISE Workshops 2007. LNCS, vol. 4832, pp. 272–283. Springer, Heidelberg (2007)
16. Preist, C.: A conceptual architecture for semantic web services. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 395–409. Springer, Heidelberg (2004)
17. Shostack, G.L.: How to Design a Service. European Journal of Marketing 16(1), 49–63 (1993)
18. Nayak, N., Nigam, A.: Modeling Business Services for Implementing on Global Business Services Delivery Platforms. In: 4th IEEE International Conference on Enterprise Computing, E-Commerce and E-Services (CEC-EEE 2007), Tokyo, Japan, pp. 577–583 (2007)
19. Euzenat, J., Shvaiko, P.: Ontology Matching. Springer, Heidelberg (2007)
20. Pryke, A., Mostaghim, S., Nazemi, A.: Heatmap visualization of population based multi objective algorithms. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 361–375. Springer, Heidelberg (2007)

Realizing Process Modifications in Container Terminals with SOA – A Prototype

Thomas Will and Thorsten Blecker

Hamburg University of Technology (TUHH),
Schwarzenbergstr. 95, 21073 Hamburg, Germany
publications@thomas-will.eu, blecker@ieee.org

Abstract. This paper deals with changes in existing IT systems resulting from RFID-based process modifications in maritime container logistics. The article demonstrates how a SOA enables this IT adoption by functioning as a software layer between process and legacy system to enable flexible processes. In addition, the paper shows how we applied the service principles during the design and the implementation phase and perform the implementation by developing a physical miniature prototype to visually show how the modifications can easily be implemented by service-based computing.

Keywords: Service-based Computing, Legacy Systems, RFID, Container Logistics, Process Modifications, Prototype.

1 Introduction

Container logistics is a growing industrial sector. Container terminals have been realizing average annual growth rates up to 11 percent since 1995. Current and expected volumes overstrain the existing infrastructure. Therefore, responsible operators are searching for new technologies to accelerate existing processes by reducing errors, manual intervention and redundancies. Radio Frequency Identification (RFID), lately standardized for container logistics by the International Organization for Standardization (ISO) [1], is such a technology. The ISO standardized three different transponders for maritime containers [2]: the license plate [3] [4], which stores container specific data, the shipment tag [5], which stores cargo / shipment specific data, and the electronic seal [6-11], which serves as a mechanical seal covering some additional features.

Introducing RFID to container logistics means integrating this technology into existing logistics processes. This paper illustrates how resulting changes in the IT infrastructure (while retaining existing IT systems) can be covered by service-based computing and how we employed service design principles during our SOA design.

To give an overview, the next chapter briefly explains the general process structure followed by the example of a transhipment process. Chapter three deals with the service design in this example and illustrates how service design principles have been applied in our scenario. Finally, chapter four provides details on the developed prototype and further activities.

E.M. Maximilien et al. (Eds): ICSOC 2010 Workshops, LNCS 6568, pp. 253–264, 2011.

2 Process Analysis and Modifications

RFID technology itself does not provide benefits unless "interacting" with the business process [12]. Thus, this section analyses the process steps a container takes while being transported from a depot to a shipper and via an intermodal transport chain to a consignee and back to another depot. It should be noted that each container transport is different, and this reference process can only illustrate a generic form of existing processes. The objective of the accomplished analysis is to detect the interdependences between material flow (container) and control flow (information) in order to provide a basis for the service design.

2.1 Research Methodology

The process analysis is based on expert consultations supported by *the Logistics-Initiative Hamburg* [13], as well as expert interviews with logistics service providers. The Logistics-Initiative Hamburg, founded by the Hamburg Business Development Corporation [14], is a registered association of companies and institutions working in the Hamburg metropolitan region. It aims to develop and expand the area of Hamburg as a logistics hub. The members decided to launch a working group to investigate the use of RFID in container logistics. This group consists of 20 experts working for logistics infrastructure providers (e.g. terminal operator), ocean carriers, logistics service providers, insurance companies and IT consultancies. In fact, there has been a core group of four experts (a terminal operator, a logistics service provider, an ocean carrier and an IT service provider) which provided the main part of relevant details for the analysis. The process analysis follows Becker's seven step waterfall model [15]: (1) define team objectives, (2) assemble team, (3) define process, (4) gather information, (5) record process, (6) document process and (7) validate process.

2.2 Process Structure

Transport, transhipment and inventory processes are the core elements of a transport chain. During the transport process, a container is loaded on a truck (T1), railway wagon (T2) or in ship tonnage (T3). During the process analysis, we focused on the transhipment processes to identify useful readout points and process modifications. The reference process (Fig. 1) summarizes the 14 investigated transhipment processes.

To give an example, the following sections explain sub-process *no. 09 – Deliver via oversea vessel* and deal with the process modifications. Sub-process no. 09 is chosen, (1) because of its simplicity, which makes it easily understandable and extendable, (2) its high automation degree, i.e. proven RFID-based enhancements in this process are more difficult than in other processes, and (3) its relevance, because unloading containers from oversea vessels is a major bottleneck in current transport chains.

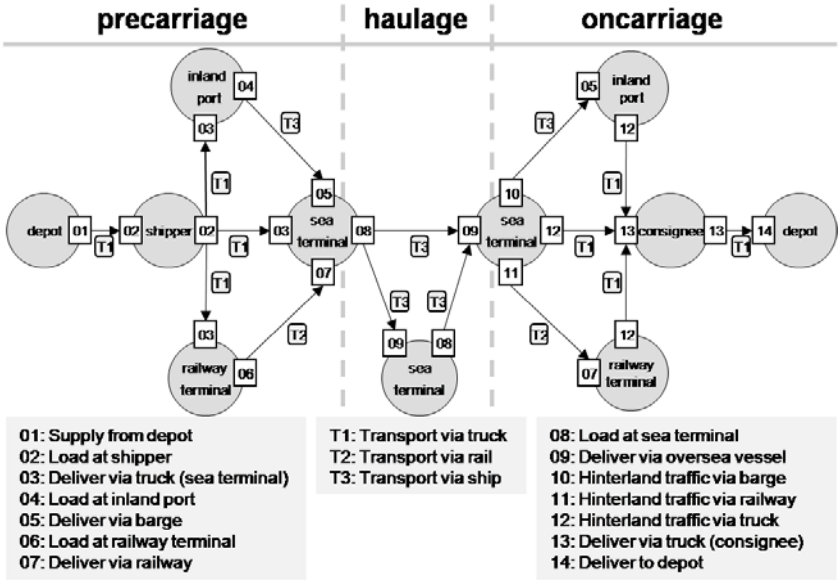


Fig. 1. Transhipment processes in intermodal container logistics

2.3 Analysis and Modifications – Sub-Process No. 09 – Deliver via Oversea Vessel

Phase 1 - data reception: As soon as the maritime carrier loads a container on his vessel, he informs the destination terminal on the container arrival. Just before the ship arrives, the maritime carrier submits the ship's stowage plan via BAPLIE[1] [16] [17], including the position of all loaded containers. Based on the advanced notification and the stowage plan, the sea terminal creates the unloading plan, which contains the sequence of transhipment processes to unload desired containers.

Phase 2 – data check: After the ship has docked, the unloading process starts. The gantry crane operator raises the container to be unloaded and another staff member checks the container number. If the container number matches with the number listed in the unloading plan, then this staff member sends the signal for dumping the container on the quay wall to the gantry crane operator. Other staff members remove the secure locks, check the container number once again and the availability of a seal, but not the seal number. Subsequent to the comparison, the data is stored in the system. In case of divergences during the checks, the container is handled manually.

Phase 3 – data forwarding: The container is stocked and the handling operator updates the unloading plan comprising the actual state of container unloads. Based on this, the handling operator creates a new stowage plan and submits it to the maritime carrier and to the next handling operator. The maritime carrier briefs the 3PL (third-party logistics provider) on the container arrival, which enables the 3PL to further

[1] BAPLIE means bay plan/stowage plan occupied and empty locations message.

specify the transport. The maritime carrier also accounts the container transport, and the 3PL creates transport orders for the oncarriage.

After developing the generic reference process, we identified the required modifications and those changes which allow enhancing the process when introducing RFID. Thus, the modifications are divided into three categories: (1) modifications that have to be accomplished to integrate RFID into the process, (2) those that accelerate the process, and (3) those that aim to extend the process functions without increasing lead time.[2] After providing an overview on the example process and the RFID-caused changes, the next chapter will explain the modifications and the resulting SOA design in more detail.

3 SOA Design

Introducing RFID to container logistics and integrating this technology into existing logistics processes leads to several process modifications; which results in IT infrastructure changes. This chapter demonstrates how a SOA facilitates this IT adoption by functioning as a software layer between process and legacy system to enable flexible processes.

Due to the fact that most of the different transhipment processes contain similar process steps that are currently implemented by a legacy application logic, the first step is to map the reusable legacy application logic into services to allow the reusability in the modified processes. Compounding these basic services that wrap the existing application logic and with additional services that provide simple and reusable utilities to the modified process, we create a basic service layer containing all legacy and newly developed basic applications. Following the naming of Thomas Erl [20], we label this layer as application service layer. Based heron, we develop services that compose available application services to execute the business logic required by the processes. These services are called value-added business services are located in the so called business service layer [20]. Finally, we developed a third service layer ontop of the existing layers that realized a workflow management by orchestrating the developed business services and enabling a mapping of the process logic via business service interaction. This topmost layer is called orchestration service layer [20].

We design the services regardless of the underlying IT system, because IT systems differ from terminal to terminal, i.e. the designed SOA is able to cope with all process steps, including the modifications without further support from an existing terminal IT system. Thus, depending on the terminal and its IT system, some of the developed services are not necessary, and can be replaced by services using the legacy system's function. The following sub-chapters first explain only those (13 out of 35) services that are used in sub-process no. 09 and second deal with the application of service design principles.

[2] Due to space restrictions, we skip the illustration of the "sub-process no. 09 – deliver via oversea vessel (process)". For an illustration of the sub processes, please consult [18]. For more detailed information on the accomplished process analysis and modification, please consult [19].

3.1 Designing Services– Sub-Process No. 09

The process-service connection are visualized with dashed rectangles which represent the newly designed services (Fig. 2). Sub-process no. 09 starts when the maritime carrier submits information about the arriving container to the sea terminal. The information consists of the container's registration and the stowage plan. For documentation purposes and because of the different submission times, the handling operator has to store both. Thus, we add two services *store container registration* and

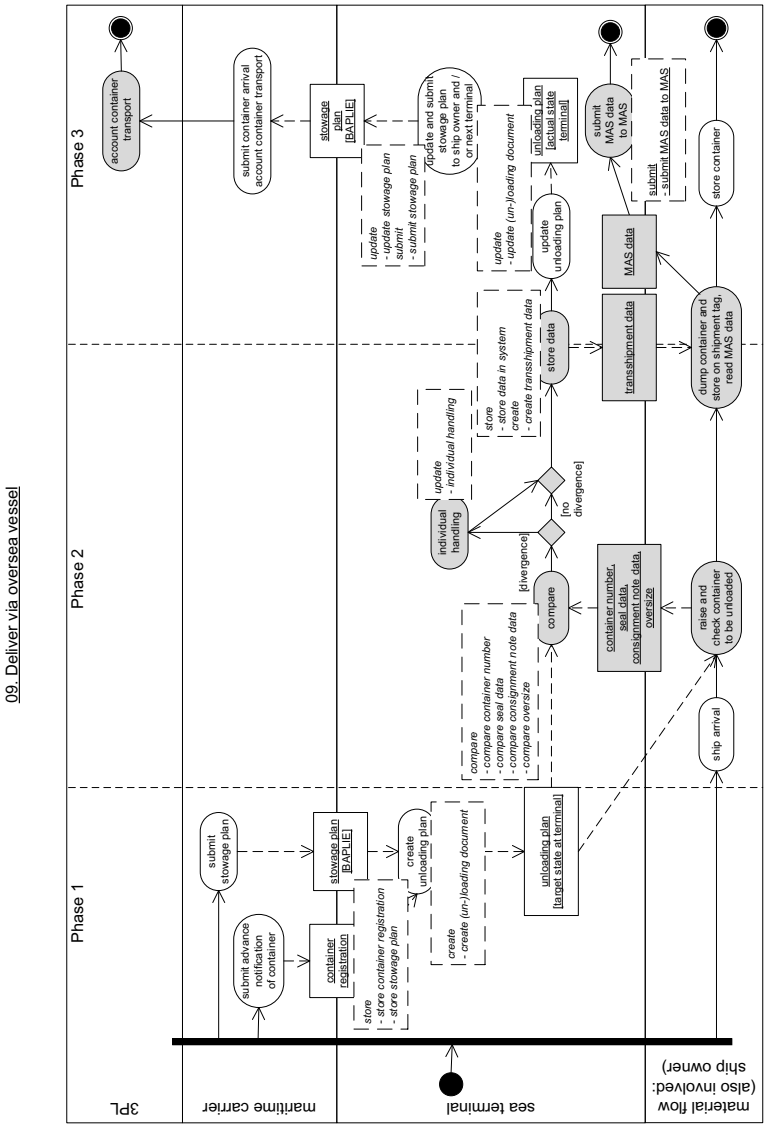


Fig. 2. Sub-process no. 09 – deliver via oversee vessel (services)

store consignment note data on the application service layer, which are compounded by the business service *store*.

After storing received information, the sea terminal has to create an unloading plan, which is realized by the application service *create (un-)loading document*. This service is covered by the business service *create*.

Today, the data comparison is a manual (paper-based) task. Using RFID requires the implementation of appropriate application services to realize above described modifications. The first process step in phase 2 contains several comparisons that are mapped to the corresponding service (in brackets): container number (*compare container number*), seal data (*compare seal data*), consignment note data (*compare consignment note data*) and oversize (*compare oversize*). A business service named *compare* comprises these application services. A divergence between the data stored in the IT system and the data stored on the transponder leads to individual handling (*individual handling*). This service provides an interface for human intervention and updates the system data, based on the human input. This application service is part of the business service *update*.

The system further stores the data (*store data in system*), creates the transhipment data and stores it on the transponder (*create transhipment data* and *store data on shipment tag*). Both application services are part of the business service *store*. After initiating the transhipment, the unloading plan and subsequently the stowage plan are updated (*update (un-)loading document* and *update stowage plan).* Finally, the terminal system submits the stowage plan to the maritime carrier (*submit stowage plan* – part of business service *submit*). Table 1 gives a brief overview on the services used in sub-process no. 09.

Table 1. Description of services used in sub-process no. 09

Service Name	Input (Source)	Output	Description
store container registration	container registration (3PL or maritime carrier)	finish /error notification	store input in the SOA
store stowage plan	stowage plan (ship forwarding agent or maritime carrier)	finish /error notification	store input in the SOA
store data in system	transshipment data (SOA) [3]	finish /error notification	store input in the SOA
store data on shipment tag	various inputs (SOA)[4]	finish /error notification	store input on shipment tag
compare container number	(un-)loading document (SOA) container number (RFID reader)	(non-)conformance notification	compare inputs
compare seal data	(un-)loading document (SOA) seal data (RFID reader)	(non-)conformance notification	compare inputs

[3] The here designed terminal SOA provides the data either by using functions from the existing legacy system (such as data storage) or by generating the data based on own computations.

[4] Depending on the process, the input is a combination of: consignment note (former paper-based), container number, eSeal number, customs declaration number (former paper-based), transhipment data (data of participants that were in contact with the container), truck transport information.

Table 1. *(continued)*

compare oversize	(un-)loading document (SOA) oversizes (RFID reader)	(non-)conformance notification	compare inputs
create (un-) loading document	various inputs (SOA) [5]	(un-)loading document[6]	match the stored input data to an (un-) loading plan and store it in the system
create transshipment data	(un-)loading document environment data[7]	transshipment data	create a data set that includes all necessary data of the transshipment.
update (un-)loading document	(un-)loading document (SOA) data of transhipped containers (RFID reader)	(un-)loading document	update (un-)loading document (based on an existing (un-)loading document) and data of transhipped containers
update stowage plan	stowage plan (SOA) (un-)loading document (SOA)	stowage plan	create a new stowage plan by updating the initial stowage plan with the accomplished (un-)loading document
individual handling	(un-)loading document (SOA) error notification (SOA) non-conformance notification (SOA)	(un-)loading document	solve the error / non-conformance by manual intervention
submit stowage plan	stowage plan (SOA)	finish /error notification	submit stowage plan to ship forwarding agent / maritime carrier

3.2 Applying Service Design Principles

After describing the developed application and business services for sub-process no. 09 in detail, we demonstrate how we applied design principles for the whole SOA to be able to accomplish the asked duties. Services have to be reusable, loosely coupled, composable, autonomous, stateless, discoverable, location transparent, share a formal contract, abstract the underlying logic, and have a network-addressable interface.

[5] The application service *create (un-)loading document* uses all data that was stored in previous process steps of this sub-process. Depending on the process, the input is a combination of: release order, container registration, customs clearance, consignment note data, sequence of wagons, stowage plan, registration stop, damaging, or truck license plate number.

[6] The service creates an unloading document if it is implemented in sub-process 3, 5, 7, 9, or 11 and a loading document if it is implemented in sub-process 4, 6, 8, 10, or 12. For sub-process 3 and 12, the list contains only one container. In these sub-processes, the (un-)loading plan is called shipment data.

[7] Environment data can include timestamp of the transhipment, identifier of the terminal and possibly data of the personnel who transhipped the container and the company that delivered the container to the terminal.

Services are reusable. Regardless of whether immediate reuse needs exist, services are designed to be potentially reusable. To fulfill this requirement services are designed generic, i.e. processing-specific logic is stored in exchanging messages. The service computes the result based on the given input and can therefore provide a general capability which can easily be reused. Table 2 focuses on the reuse of developed application services in the modified sub-processes 03 to 12 and illustrates the high reuse-rate of designed services (# displays the amount of reuse per service).

Table 2. Reuse of services used in sub-process no. 03 to 12

store services\ sub-process no.	3	4	5	6	7	8	9	10	11	12	#
store release order								X	X	X	3
store container registration	X		X		X		X				4
store customs clearance		X		X		X		X	X	X	6
store consignment note data	X	X	X	X	X	X	X	X	X	X	12
store sequence of wagons				X	X				X		3
store stowage plan		X	X			X	X	X			5
store registration stop		X				X					2
store damaging	X	X		X	X	X		X	X	X	8
store truck license plate number	X									X	2
store data in system	X	X	X	X	X	X	X	X	X	X	12
store data on shipment tag	X	X	X	X	X	X	X	X	X	X	12
compare services\ sub-process no.	3	4	5	6	7	8	9	10	11	12	#
compare sequence of wagons				X	X				X		3
compare position on wagon number				X	X				X		3
compare container number	X	X	X	X	X	X	X	X	X	X	12
compare consignment note	X		X		X					X	4
compare customs clearance										X	1
compare seal data	X	X	X	X	X	X	X	X	X	X	12
compare oversize	X		X		X		X				4
compare dangerous cargo label (yes/no)	X	X		X	X	X		X	X	X	10
compare truck license plate number										X	1
create services\ sub-process no.	3	4	5	6	7	8	9	10	11	12	#
create (un-)loading document	X	X	X	X	X	X	X	X	X	X	12
create shunting tasks				X					X		2
create interchange	X										1
create receipt										X	1
create transshipment data	X	X	X	X	X	X	X	X	X	X	12
update services\ sub-process no.	3	4	5	6	7	8	9	10	11	12	#
update (un-)loading document	X	X	X	X	X	X	X	X	X	X	12
update sequence of wagons				X	X				X		3
update stowage plan		X	X			X	X	X			5
update missing data	X				X					X	3
update consignment note data	X									X	2
individual handling	XX	X	X	XX	X	X	X	X	XX	X	15
submit services\ sub-process no.	3	4	5	6	7	8	9	10	11	12	#
submit sequence of wagons				X	X				X		3
submit stowage plan		X	X			X	X	X			5
submit consignment note data	X										1
submit status information										X	1

Services are loosely coupled. We established a loosely coupled relationship between services, their underlying application logic, respectively. The services are connected via formal contracts to each other, enables a coupling as long as this specific orchestration of operations / services is required. The service reusability illustrated in Table 2 further indicates that the services are loosely coupled, because they are reused in diverse sub-processes.

Services share a formal contract. In order to interact with each other, the services share specific information such as the service result, input and output message as well as rules and characteristics of the service and its operations. We use the XML specification WSDL [21], an open standard for these service descriptions.

Services abstract underlying logic. The visible part of a service is what is exposed via the service's description and formal contract. As we use the service-oriented computing as an intermediate layer between legacy system and process, the underlying logic is completely invisible and irrelevant to service consumers and there is no restriction concerning the size, allocation, distribution or complexity of the underlying application logic.

Services are composable. This principle ensures that our services can reuse other services to accomplish their work and do not need to redundantly implement specific operations. The service itself can also be part of other service compositions, which use its function to accomplish the own goals.

Services are autonomous. The underlying logic that is governed by a service works within an explicit boundary. For execution, this logic is not dependent on other applications outside the service's autonomy area. This does not necessarily mean, that a service has exclusive ownership of the encapsulated logic, but only guarantees control over underlying logic at the time of execution. As the existing legacy system functions are triggered by external inputs, mainly coming from manual data entries, we simply replace these manual inputs with RFID-reads, forwarded by the SOA and a service, respectively. Thus we can assure that the service has control over the legacy function at the time of execution.

Services are stateless. Statelessness is a required design principle to assure loosely coupled services and promote reusability. Our services minimize the amount of state information they manage and the duration they hold it. In fact, all services are stateless and receive all relevant state information via the incoming trigger-messages.

Services are discoverable. Each service provides a piece of processing logic, which might potentially be reusable. To prohibit redundant creation of services and underlying logic, being discoverable is an important service design criteria. Each service provides descriptions (WSDL) to be discovered and understood by humans as well as service consumers who may be able to make use of the services' logic.

Services have a network-addressable interface. Service requestors must be able to invoke a service across the network. Each service can be executed via an internal interface as well as via HTTP and can return the results via both ways.

Services are location transparent. Service consumers do not have to access a service using its absolute network address. They dynamically discover the location by simply submitting their request to a registry that triggers the corresponding service. This feature allows services to move from one location to another without affecting the consumers.

4 Prototype and Further Work

After designing the required services, we currently develop a real-world prototype that implements them. The prototype's goal is to illustrate the applicability of service oriented computing in the area of maritime container logistics to give logistics practitioners an understanding of this technology. Thus, the prototype is divided in two parts: (1) the service oriented computing discussed in this paper and (2) a physical miniature prototype of a container terminal and transport vehicles (ship, train, truck) mapping the above presented processes.

Fig. 3 provides a schematic overview on the container terminal, while Fig. 4 shows the real prototype implementation.

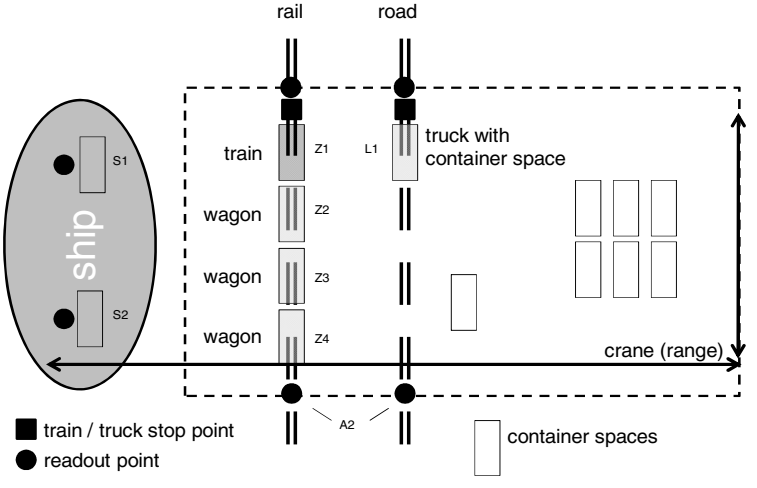


Fig. 3. Physical miniature prototype – schematic terminal view

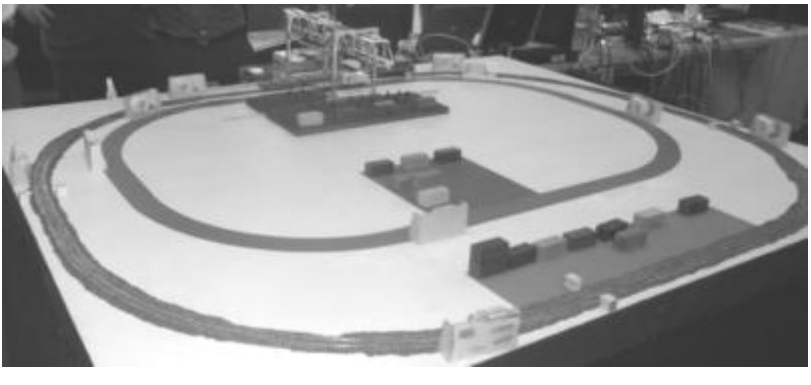


Fig. 4. Physical miniature prototype – real implementation view

Besides implementing the terminal-internal processes with the SOA, the prototype should also focus on cross-company processes. Thus, the physical prototype also contains two customer areas to map transport chain processes (Fig. 5).[8]

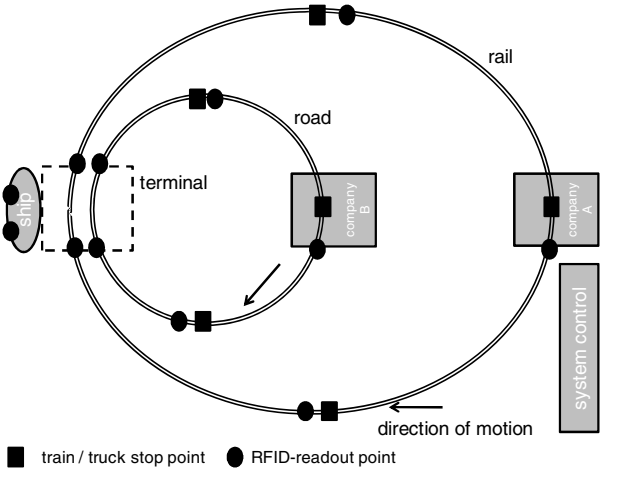


Fig. 5. Physical miniature prototype – schematic overview

The idea is to provide a handy example to the practitioners along the logistic transport chain and to map cross-company processes. This includes tracing and tracking of goods and transport vehicles, exception handling in case of unexpected behaviours of transport vehicles and containers. The goal is also to demonstrate how service oriented computing facilitates cross-company communication (e.g. with automated messages via email/HTTP/SOAP interfaces), increases the amount of valuable information in the transport chain and the ability to easily change processes for different (or the same) customer. SOA further allows developing a high-level business process and defining inputs and outputs for every process step to split a big process in small entities that can be implemented by wrapping service - that wraps legacy system functions - or utility services - that implement small parts of the business logic and are potentially reusable.

References

1. International Organization for Standardization, `http://www.iso.org/iso/home.htm`
2. DIN ISO 668, Series 1 Freight Containers - Classification, Dimensions and Ratings (October 1999)
3. ISO 10374.2, Freight Container – Automatic Identification (2009)
4. ISO 6346, Freight Containers – Coding, Identification and Marking (1995)

[8] Note: This second development phase starts in October 2010 and is planned to be finished within two months.

5. ISO 18186, Freight containers – RFID cargo shipment tag (2010)
6. ISO 17712, Freight Containers – Mechanical Seals (2006)
7. ISO 18185-1, Freight Container – Electronic Seals – Part 1: Communication Protocol (2007)
8. ISO 18185-2, Freight Container – Electronic Seals – Part 2: Application Requirements (2007)
9. ISO 18185-3, Freight Container – Electronic Seals – Part 3: Environmental Characteristics (2007)
10. ISO 18185-4, Freight Container – Electronic Seals – Part 4: Data Protection (2007)
11. ISO 18185-5, Freight Container – Electronic Seals – Part 5: Physical Layer (2007)
12. Hellström, D.: The cost and process of implementing RFID technology to manage and control returnable transport items. International Journal of Logistics Research and Applications 12(1), 1–21 (2009)
13. Logistics-Initiative Hamburg, http://www.hamburg-logistik.com
14. HWF – Hamburg Business Development Corporation, http://www.hamburg-economy.de/index_en.html
15. Becker, T.: Prozesse in Produktion und Supply Chain Optimieren. Springer, Berlin (2005)
16. UN/EDIFACT Message BAPLIE Release: 08A, Bayplan/stowage plan occupied and empty locations message, http://www.unece.org/trade/untdid/d08a/trmd/baplie_c.htm
17. UN/EDIFACT List of all Messages Release, http://www.unece.org/trade/untdid/d00a/trmd/trmdi2.htm
18. Will, T., Blecker, T.: Precarriage in Container Logistics – Analysis and RFID-Driven Modifications in Transshipment Processes. In: 18th IPSERA Conference: Supply Management – Towards an Academic Discipline?, pp. 1554–1569 (2009)
19. Will, T.: Creating a Dynamic Speech Dialogue: How to implement dialogue initiatives and question selection strategies with VoiceXML agents. VDM Verlag Dr. Müller (2007)
20. Erl, T.: Service-Oriented Architecture – Concepts, Technology, and Design. Prentice Hall PTR, Upper Saddle River (2005)
21. W3C: Web Service Definition Language (WSDL), http://www.w3.org/TR/wsdl

Author Index