Petri Mähönen
Klaus Pohl
Thierry Priol (Eds.)

LNCS 5377

# Towards a Service-Based Internet

First European Conference, ServiceWave 2008
Madrid, Spain, December 2008
Proceedings

Springer

Lecture Notes in Computer Science 5377

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

Editorial Board

David Hutchison
*Lancaster University, UK*
Takeo Kanade
*Carnegie Mellon University, Pittsburgh, PA, USA*
Josef Kittler
*University of Surrey, Guildford, UK*
Jon M. Kleinberg
*Cornell University, Ithaca, NY, USA*
Alfred Kobsa
*University of California, Irvine, CA, USA*
Friedemann Mattern
*ETH Zurich, Switzerland*
John C. Mitchell
*Stanford University, CA, USA*
Moni Naor
*Weizmann Institute of Science, Rehovot, Israel*
Oscar Nierstrasz
*University of Bern, Switzerland*
C. Pandu Rangan
*Indian Institute of Technology, Madras, India*
Bernhard Steffen
*University of Dortmund, Germany*
Madhu Sudan
*Massachusetts Institute of Technology, MA, USA*
Demetri Terzopoulos
*University of California, Los Angeles, CA, USA*
Doug Tygar
*University of California, Berkeley, CA, USA*
Gerhard Weikum
*Max-Planck Institute of Computer Science, Saarbruecken, Germany*

Petri Mähönen   Klaus Pohl
Thierry Priol (Eds.)

# Towards a Service-Based Internet

First European Conference, ServiceWave 2008
Madrid, Spain, December 10-13, 2008
Proceedings

Springer

Volume Editors

Petri Mähönen
RWTH Aachen University, Department of Wireless Networks
Kackertstrasse 9, 52072 Aachen, Germany
E-mail: pma@mobnets.rwth-aachen.de

Klaus Pohl
University Duisburg-Essen, Software Systems Engineering
Schützenbahn 70, 45117, Essen, Germany
E-mail: pohl@sse.uni-due.de

Thierry Priol
INRIA Rennes - Bretagne Atlantique
Campus de Beaulieu, 35042 Rennes Cedex, France
E-mail: thierry.priol@inria.fr

Library of Congress Control Number: 2008940580

CR Subject Classification (1998): K.6, K.8, K.4.4, H.4, H.3.5

LNCS Sublibrary: SL 3 – Information Systems and Application, incl. Internet/Web and HCI

| | |
|---|---|
| ISSN | 0302-9743 |
| ISBN-10 | 3-540-89896-4 Springer Berlin Heidelberg New York |
| ISBN-13 | 978-3-540-89896-2 Springer Berlin Heidelberg New York |

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

Springer is a part of Springer Science+Business Media

springer.com

© Springer-Verlag Berlin Heidelberg 2008
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12581429 06/3180 5 4 3 2 1 0

# Foreword

Today it is almost impossible to remember what life was like with no computer, no mobile phone, and no Internet for accessing information, performing transactions or exchanging emails and data. New technology is bringing wave after wave of new benefits to daily life: organisations are doing business with each other via the Internet; people are filling in tax declarations online and booking their next vacation through the Internet. In general we are all progressively using (and dependent on) software and services running on computers, connecting mobile phones and other devices, and exchanging information on the Internet.

People like to shop around and exercise choice. So do businesses and public administrations. Today they can buy a complete software package that best suits their needs, even though they may never use some of the tools it offers, or other desirable tools are not available. In the future they may no longer have to compromise on choice. Alternative approaches like "Software as a Service" and "Computing Resources as a Service" are emerging. Software is provided on-line as a service when and where it is needed, and the same for computing resources needed to run software. Such an approach allows individuals and organisations to tap into and effectively harness the immense wealth of information, knowledge and analytical resources when they need them, paying only for what they use. Customers are bound to benefit when there is a sufficiently rich choice of services.

But what does this mean when seen from the supply side? Although it may not yet be so evident to the outside world, the software industry is rapidly restructuring and changing patterns of competition and business. Relatively new internet companies are successfully providing software and computing resources as a service. They are seriously competing with established industries delivering traditional packaged software, forcing the latter to adapt their business models.

I believe that the restructuring of the market and the industry represents a tremendous opportunity for Europe and for the European software industry.

The number of companies and researchers involved in the European Technology Platforms eMobility, EPoSS, ISI, NEM and NESSI, shows that industry is also taking this opportunity seriously. These Technology Platforms are collaborating to define a strategy for addressing the challenges in delivering software and computing resources as a service, and in developing new attractive context-based, device-independent services addressing the precise needs of the user that exploit the possibilities offered by the convergence of the media, telecom and IT industries.

ServiceWave 2008 presents the main results of European industrial and academic collaboration in this field. It reflects the great potential and the continued strength of European expertise in many fields of service development. The challenge now is to bring these advancements to the market and to ensure the European leadership in the new Internet economy.

Viviane Reding
Commissioner for Information Society and Media

# Preface

The Internet is becoming a critical infrastructure for the growth of the modern economy and society by allowing anyone to access information in a transparent way, independent of the location or computing device used. While today, users' main interactions with the Internet remain web browsing or exchange of e-mails, a revolution is on-going through the availability of software services.

The convergence of smarter networks, resource virtualization and collaborative and intelligent services will pave the way to the Future Internet with new software services which will be as revolutionary as the e-mail and the Web were when they appeared initially.

The Future Internet will enable new and unforeseen collaborative applications based on dynamic adaptation to the context and to the available services and resources. The technological and architectural advances in the Internet domain will also take economic and service-oriented facts into account.

Furthermore, the Future Internet will reinforce the need for interdisciplinary collaboration between domains such as computer science, electrical engineering and social sciences - disciplines that have already moved forward in working together but whose level of collaboration will have to increase considerably. However, before this revolution can be completed, several research challenges must be addressed and a tighter collaboration between industry and academia is of utmost importance.

The ServiceWave conference series aims to establish the premier European forum for researchers, educators and industrial practitioners to present and discuss the most recent innovations, trends, experiences and concerns in software services (or the "Future of the Internet of Services") and related underlying network technologies. ServiceWave fosters the creation of cross-community scientific excellence by gathering together industrial and academic experts from various disciplines such as business process management, distributed systems, computer networks, wireless and mobile communication networks, grid computing, embedded and smart systems, networking, service science and software engineering.

ServiceWave 2008 was organized along four tracks designed to foster the collaboration and lively exchange of ideas between industry and academia:

- ETPs Future Internet Track: dedicated to presentations on the common vision of the Future of the Internet elaborated and shared by the major European Technology Platforms active in the ICT domain—namely, eMobility, EPoSS, ISI, NEM and NESSI
- Industrial Panels Track: multidisciplinary open debate panels focusing on topics which, starting from traditional service-based systems engineering, brought the discussions toward the ambitious target of Future Internet

– Scientific Track: presentation of the papers, selected by the ServiceWave 2008 Program Committee, from research and industry, covering the state of practice and real-world experiences in service engineering
– Workshop Track: meetings organized by Working Groups active within the Future Internet activities, the intra-ETP activities or within an ETP as well as research and industrial communities wishing to present their work

These proceedings cover the peer-reviewed Scientific Track of ServiceWave 2008 for which we received 102 submissions. In an extensive and careful review process, the Scientific Program Committee of ServiceWave 2008 accepted 28 papers—an acceptance rate of 27%.

We offer our sincere thanks to the members of the ServiceWave 2008 Program Committee for devoting their time and knowledge to reviewing and discussing the submitted papers. We would especially like to thank the members of the Program Committee who attended the two-day Program Committee meeting held in Essen, Germany on September 1 and 2, 2008. Special thanks go to Andreas Gehlert and Julia Hilscher for their responsive and helpful support during the paper evaluation and selection process, as well as during the preparation of the proceedings.

We would also like to thank David Kennedy for organizing the ETP Track, Stefano De Panfilis and Wolfgang Gerteis for the Industrial Track and Frédéric Gittler for the Workshop Track.

In addition, ServiceWave 2008 would not have been possible without the efforts and expertise of a number of people who selflessly offered their time and energy to help to make this conference a success. We would like to thank Universidad Politécnica de Madrid, host of the 2008 edition of ServiceWave, for their continous support.

We would like to thank all the people on the Organizing Committee, especially Federico Alvarez, Nuria Sánchez Almodóvar, Usoa Iriberri Zubiola, Bruno François-Marsal, Véronique Pevtschin and Barbara Pirillo.

Finally, we thank the main conference sponsors and supporters: Alcatel-Lucent, Atos Origin, Engineering, Siemens, Telefónica, Thales, IBBT, HP and IBM as well as INRIA, RWTH Aachen, and University of Duisburg-Essen.

December 2008

Reinhold Achatz
Guillermo Cisneros
Petri Mähönnen
Klaus Pohl
Thierry Priol

## Organization

ServiceWave 2008 was organized by NESSI and hosted in Madrid by UPM in cooperation with:

– The European Technology Platforms eMobility, EPoSS, ISI and NEM
– Spanish Technology Platform INES
– Networks of Excellence CoreGrid and S-Cube
– Support actions Eiffel, 4NEM and NESSI 2010
– ICSOC, the International Conference on Service-Oriented Computing

### General Chairs

| | |
|---|---|
| Reinhold Achatz | Siemens, Germany |
| Guillermo Cisneros | Universidad Politécnica de Madrid, Spain |

### Program Chairs

| | |
|---|---|
| Petri Mähönnen, Coordinator of EIFFEL | RWTH Aachen University, Germany |
| Klaus Pohl, Coordinator of S-Cube | University Duisburg-Essen, Germany |
| Thierry Priol, Coordinator of CoreGRID | INRIA, France |

### Program Committee

| | |
|---|---|
| Federico Alvarez | Universidad Politécnica Madrid, Spain |
| Mike Boniface | University of Southampton, UK |
| Vincent Boutroux | France Telecom, France |
| Jose Maria Cavanillas | ATOS Origin SA, Spain |
| Costas Courcoubetis | Athens University of Economics and Business, Greece |
| Marco Danelutto | University of Pisa, Italy |
| Stefano De Panfilis | Engineering I.I., Italy |
| Eric Dubois | CRP Henri Tudor, Luxembourg |
| Serge Druais | Thales, France |
| Schahram Dustdar | Vienna University of Technology, Austria |
| Frank Fitzek | University of Aalborg, Denmark |
| Paraskevi Fragopoulou | FORTH, Greece |
| Vladimir Getov | University of Westminster, UK |
| Carlo Ghezzi | Politecnico di Milano, Italy |

| | |
|---|---|
| Sergei Gorlatch | University of Muenster, Germany |
| Pierre Guisset | CETIC, Belgium |
| Andrei Gurtov | HIIT, Finlande |
| Mohand-Said Hacid | University of Lyon, France |
| Stephen Hailes | University College London, UK |
| Manuel Hermenegildo | Universidad Politécnica Madrid, Spain |
| Paola Inverardi | University of L'Aquila, Italy |
| Borka Jerman-Blazic | Jozef Stefan Institute, Slovenia |
| Peter Kacsuk | MTA SZTAKI, Hungary |
| Roger Kilian-Kehr | SAP, Germany |
| Domenico Laforenza | ISTI-CNR, Italy |
| Frank Leymann | University of Stuttgart, Germany |
| Neil Maiden | City University London, UK |
| Katharina Mehner | Siemens AG, Germany |
| Andreas Metzger | University of Duisburg-Essen, Germany |
| Norbert Meyer | Supercomputing Center, Poland |
| Werner Mohr | Nokia Siemens Networks, Germany |
| Christos Nikolaou | University of Crete, Greece |
| Evgeny Osipov | LTU, Sweden |
| Jörg Ott | TKK Helsinki, Finland |
| Dimitri Papadimitriou | Alcatel-Lucent, Belgium |
| Mike Papazoglou | Tilburg University, The Netherlands |
| Jean-Louis Pazat | INSA-Rennes, France |
| Ron Perrott | Queen's University of Belfast, UK |
| George Polyzos | AUEB, Greece |
| Janne Riihijärvi | RWTH Aachen University, Germany |
| Santi Ristol | Chairman of INES, Spain |
| Ita Ritchardson | Lero, University of Limerick, Ireland |
| Colette Rolland | University Paris 1, France |
| Ian Sommerville | St. Andrews University, UK |
| Domenico Talia | Università della Calabria, Italy |
| Paolo Traverso | FBK, Trento, Italy |
| Dirk Trossen | British Telecom, UK |
| Klaus Wehrle | RWTH Aachen, Germany |
| Ramin Yahyapour | University of Dortmund, Germany |
| Wolfgang Ziegler | Fraunhofer SCAI, Germany |

## Referees

| | | |
|---|---|---|
| I. Aktas | M. Carro | O. Danylevych |
| M. Autili | J.-M. Cavanillas | P. Dazzi |
| A. Basukoski | A. Charfi | J. Duennweber |
| E. Bertin | C. Comito | H. Eberle |
| D. Bianculli | M. Coppola | C. Ghezzi |
| C. Cordier | C. Cunningham | R. Giaffreda |

F. Glinka
M.-C. Gomes
K. Görlach
E. Grandry
A. Gurtov
T. Harmer
J. Hawkins
T. Heer
P. Inverardi
D. Ivanovic
G. Jankowski
M.C. Jäger
B. Jerman-Blazic
R. Krishnaswamy
P. Kegel
J. Koskela
S. Kubicki
G. Kunz
O. Landsiedel
H. Li
R. Lichwala
J.A. Bitsch Link
P. Lopez-Garcia
K. Loughran
D. Malandrino
C. Mastroianni
W. Mohr
G. Monakova
J. Navas
E. Di Nitto
S. Nurcan
M. Owsiak
P. Pelliccione
G. Peter
E. Pfeuffer
G. Pirrò
M. Plociennik
A. Ploss
A. Raed
F. Raimondi
S. Ramel
M. Reiter
D. Di Ruscio
R. Sasnauskas
D. Schumm
L. Serafini
A. Spriestersbach
Y. Taher
M. Tivoli
G. Tolomei
I. Trajkovska
P. Trunfio
T. Unger
K. Voigt
E. Weingaertner
S. Wieczorek
P. Wright
E. Zumpano

# Table of Contents

# An Integrated Approach for the Run-Time Monitoring of BPEL Orchestrations*

Luciano Baresi[1], Sam Guinea[1], Raman Kazhamiakin[2], and Marco Pistore[2]

[1] Politecnico di Milano – Dipartimento di Elettronica e Informazione, Italy
{baresi,guinea}@elet.polimi.it
[2] Fondazione Bruno Kessler – IRST, Trento, Italy
{raman,pistore}@fbk.eu

**Abstract.** In this paper, we compare and integrate Dynamo and ASTRO, two previous approaches of the authors for run-time monitoring of BPEL orchestrations. A key element of the proposed integrated framework is the capability to cover a wide range of features, including the detection of complex behavioural patterns, the possibility to measure boolean, numeric and time-related properties, the possibility to monitor the behaviour of the composition both at the level of a single execution instance and by aggregating the information of all execution instances of a given composition.

## 1 Introduction

BPEL (Business Process Execution Language) is the most widely used solution for workflow based cooperation amongst web services. The distributed nature of BPEL processes, the absence of a single stakeholder, the fact that partner services can dynamically change their functionality and/or QoS, and the possibility to define abstract processes and look for actual services at run time, preclude design- time validation of such systems. The reliability and robustness of these systems must be enforced by means of defensive programming techniques and suitable monitoring of executions, in order to detect problems and trigger recovery activities. BPEL supports primitive forms of probing (e.g., timeouts) and exception handling, but these features are not as powerful and flexible as needed. For this reason, several approaches have been proposed in literature for specifying monitoring directives externally to the BPEL processes and for supporting the run-time monitoring of these directives – see for instance [2,3,4,5,6,7,8,9,10,11]. Among all these different alternatives, we focus here on two approaches developed by the authors of this paper, namely Dynamo [5,6] and ASTRO [2,3]. Even if both approaches address the problem of the run-time monitoring of BPEL processes, the developed solutions are rather different. Indeed, in [5,6] the focus is on the specification of monitoring directives that can be activated and de-activated for each process execution, according to the user's preferences; the actual monitoring of these directives is performed by weaving them into the process they belong to. In [2,3], instead, the focus is on the specification of properties which may span over multiple executions of BPEL

* 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).

© Springer-Verlag Berlin Heidelberg 2008

processes and that aggregate information about all these executions; moreover the architecture clearly separates the BPEL execution engine and the monitoring engine.

In this paper, we perform a detailed comparison of the two approaches, in terms of the basic events they are able to monitor, the way these basic events can be combined to monitor complex properties, the granularity of executions that a monitor can cover (single execution vs multiple executions), and the level of integration of process execution and monitoring (e.g., tangled together or separated). The outcome of this comparison is that the two approaches have taken complementary approaches in all these perspectives, and as a consequence they have complementary strengths and weaknesses. This complementarity opens up the possibility of combining the two approaches in order to exploit the strengths of both of them. In this paper, we propose a novel approach that is obtained by integrating the two existing approaches. We describe both the language and the monitoring architecture for this new approach. A key element of the new framework is the capability to cover a wider range of features than any other monitoring approach for BPEL orchestration the authors are aware of.

The organisation of the paper is as follows. Section 2 presents a simple case study used throughout the paper. Section 3 and 4 briefly introduce Dynamo and ASTRO, while Section 5 compares them. Section 6 sketches the integrated approach resulting from combining the two aforementioned proposals. Section 7 concludes the paper.

## 2 Tele-Assistance Service

The Tele-Assistance Service (from now on `TA`) is a BPEL process that manages the remote tele-assistance of patients with diabetes mellitus. These patients have glucose meters at home that communicates with `TA`, allowing their glucose levels to be constantly monitored. Figure 1 illustrates the overall process. It uses square brackets to indicate the nature of the BPEL activity, and angular brackets to indicate the remote partner service being called. The process interacts with five partners: (a) the patient's home device (`PHD`), (b) the hospital's patient record registry (`PRR`), (c) the hospital's medical laboratory (`LAB`), (d) a pool of doctors that can provide on-the-fly home assistance (`DOC`), and (e) an ambulance emergency center (`AMB`).

A new instance of service `TA` is instantiated remotely when the client turns on his/her glucometer, which sends an appropriate `startAssistance` message. The process starts by obtaining the patient's medical records from `PRR`, in order to tell the patient (through their glucometer) the exact insulin dose to use. Once this "setup" phase is concluded, `TA` enters a loop in which a BPEL pick activity is used to gain information from the outside world. The pick activity defines three possible `[ON MESSAGE]` branches. Notice that all the decisions taken within the process are made persistent by calling the `PRR` service. The first branch, called `vitalParams`, is used to receive a periodic update from the patient's glucometer. It immediately starts by sending the data to the medical lab (`[INVOKE] analyzeData`) for an online analysis; when the results are ready they are used to decide what to do. If the patient's results are fine, no action is performed. If there are reasons to change the insulin dose being used, this is communicated to the patient (`[INVOKE] changeDose`). If the results highlight a "mild" anomaly, it is communicated to the doctors (`[INVOKE] alarm('mild')`) so that they can schedule a visit

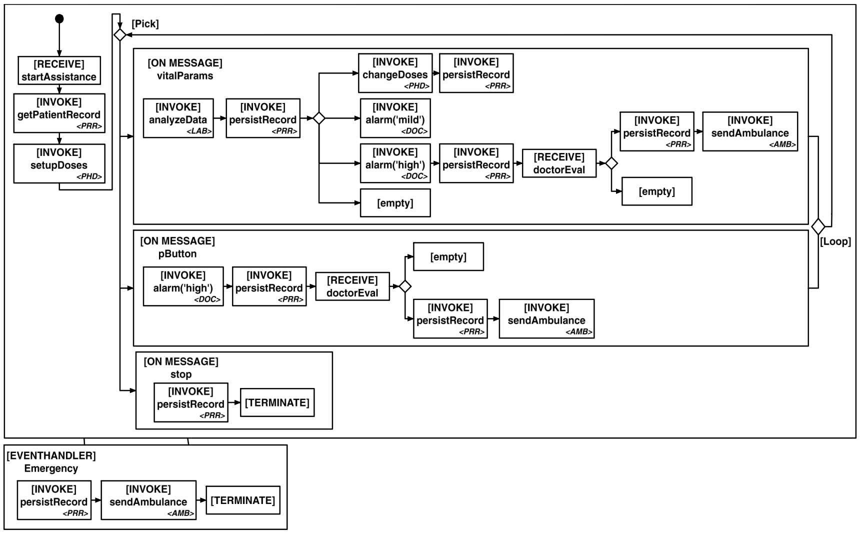

[Pick]
[RECEIVE] startAssistance
[INVOKE] getPatientRecord <PRR>
[INVOKE] setupDoses <PHD>
[ON MESSAGE] vitalParams
[INVOKE] analyzeData <LAB>
[INVOKE] persistRecord <PRR>
[INVOKE] changeDoses <PHD>
[INVOKE] persistRecord <PRR>
[INVOKE] alarm('mild') <DOC>
[INVOKE] alarm('high') <DOC>
[INVOKE] persistRecord <PRR>
[RECEIVE] doctorEval
[INVOKE] persistRecord <PRR>
[INVOKE] sendAmbulance <AMB>
[empty]
[empty]
[Loop]
[ON MESSAGE] pButton
[INVOKE] alarm('high') <DOC>
[INVOKE] persistRecord <PRR>
[RECEIVE] doctorEval
[empty]
[INVOKE] persistRecord <PRR>
[INVOKE] sendAmbulance <AMB>
[ON MESSAGE] stop
[INVOKE] persistRecord <PRR>
[TERMINATE]
[EVENTHANDLER] Emergency
[INVOKE] persistRecord <PRR>
[INVOKE] sendAmbulance <AMB>
[TERMINATE]

**Fig. 1.** The Tele-Assistance Service

to the patient. Notice that whenever the process contacts the doctors it expects an immediate acknowledgement message. If the results highlight a “serious” anomaly, it is communicated to the doctors (`[INVOKE] alarm('high')`). This causes the doctors to rush to the patient’s house and perform a complete checkup. The results are received by `TA` through a new receive activity called `doctorEval`, after which two things can happen. Either the doctors have solved the problem and the `TA` service can continue regularly, or the problem cannot be solved easily and hospitalisation is deemed necessary. In this case an order is sent to the ambulance (`[INVOKE] sendAmbulance`) so that it can go pick up the patient. The second branch, called `pButton`, represents a panic button that the patient can use to communicate an emergency. This emergency is treated exactly as in the previous thread. The third branch, called `stop`, can be used to terminate the `TA` service. This branch is called either when the patient turns off his/her glucometer, or when the ambulance picks up the patient for hospitalisation. The overall process also provides an `[EVENTHANDLER]`, which can be used at any time to require the patient’s immediate hospitalisation. It is used by the doctors, for example, to require hospitalisation in case of a “mild” alarm which turns out to be worse than expected.

In the following, we report some examples of monitoring directives that are relevant in this service. A *first property* is that the changes in the insulin doses suggested by the medical lab analysis should not vary “too” much during a short period. For instance, we may require that a suggested insulin dose should not differ from the previous value by more than 5%. Or we may require that the difference between the highest and the lowest suggested doses are within a range of 20%. A *second property* is on the performance of the service. For instance, we may want to monitor that the time it takes the doctors to send back an acknowledgement must never exceed 500ms, and that the average response time should be below 200ms. Notice that the average response time should be computed considering all the executions of the `TA` service, not just one. A *third property* is on the number of times a given event occurs: for instance, we may be interested in knowing the number of times hospitalisation has been necessary for a given patient. Notice that this property should be computed considering all the executions of the TA service for a given patient. A *fourth property* is on the temporal behaviour of the service: for instance, we may be interested in monitoring whether hospitalisation has been decided after the insulin dose has been progressively incremented for three or more times; we may also be interested in knowing the percentage of cases where such an increment in the dose has lead to hospitalisation. *Other properties* may result from the combination of the TA process described in this section with other services. For instance, if a patient is monitored with more sensors than just the glucometer (e.g., sensors for heart rate, pression, temperature...) and these sensors are managed by other services, then would be important to define monitoring properties that correlate conditions and events across these services.

## 3 Dynamo

In Dynamo [5,6] monitoring rules are made up of a location and a monitoring property. Additionally, a set of reaction strategies can be defined [6], but this is out of the scope of this paper. The *location* uses an XPath expression to select the point in the process for which we are defining monitoring, and a keyword for the “kind” of triggering condition we want (a pre- or a post-condition). Possible points of interest are for instance BPEL invoke and receive activities.

Monitoring *properties* are defined using WSCoL, an XML-aware language for the definition of behavioural properties. WSCoL defines three kinds of variables (i.e., internal, external, and historical), and expresses relationships that must hold between them. Internal variables consist of data that belong to the state of an executing process, and are defined by indicating the name of a BPEL variable (preceded by a $) and an XPATH expression that “chooses” one of the simple data values it contains (i.e., a number, a string, or a boolean). External variables consist of data that cannot be obtained from within the process, but must be obtained externally through a WSDL interface. This solution facilitates the distribution of probes and helps control the deployment of the monitoring infrastructure. This also allows for specifying certain QoS properties that can only be collected with the help of special purpose probes. Finally, historical variables are introduced to predicate on internal and external variables collected during previous activations of the monitoring framework, either from within the same process execution or from a completely different process.

In WSCoL, we can also define variable aliases. This allows us to write simpler and clearer properties, and more importantly, when we have the same external variable referenced more than once in a property, we can either collect it is as many times as needed or collect it once and define an alias for future references. This is crucial when the value of an external variable may vary depending on the exact moment in which it is collected. To define the relationships that must hold among these variables, we can use the typical boolean, relational, and mathematical operators. The language also allows us to predicate on sets of values through the use of universal and existential quantifiers, and provides a number of aggregate constructs such as max, min, avg, sum, product, and num_Of. These constructs become quite meaningful in conjunction with historical variables and allow us to compare the behaviour of a remote service with previous iterations. To better clarify how WSCoL is used, we present the following monitoring properties, defined in the context of our TA service example.

```
let $doseNew=($labResults/suggestedDose);
let $doseOld=retrieve(pID, uID, iID,
      '[INVOKE]changeDose/postcondition', '$doseStored', 1);
$doseNew <= $doseOld*1.05 && $doseNew >= $doseOld*0.95;
```

In this example we define a post-condition for [INVOKE] analyzeData, in which we state that the change in the insulin dose suggested by the medical lab analysis should not differ from the previous value by more than 5%. The property uses two variable aliases. The former (i.e., $doseNew) is defined for an internal variable responsible for extracting the new insulin dose from the BPEL variable labResults using the XPath expression /suggestedDose. The latter (i.e., $doseOld) is defined for a historical variable. The historical variable is obtained using the appropriate WSCoL retrieve function. The function takes, as parameters, the process name, the user ID, and the instance ID, allowing us to indicate that we are only interested in variables that were stored from within the process instance in execution. Its remaining parameters, on the other hand, allow us to state that we are interested in a variable that was stored in [INVOKE] changeDose's post-condition, and that was called "doseStored".

This example can be extended by stating that the difference between the highest and the lowest dosage suggestions should be within a 20% range. To do this we must calculate the maximum and minimum of the last 10 dosages that were stored:

```
let $vals=retrieve(pID, uID, iID,
      '[INVOKE]changeDose/postcondition', '$doseStored', 10);
let $min= (min $d in $vals; $d); let $max= (max $d in $vals; $d);
$min > $max * 0.80;
```

Here we use the min and max aggregate functions, which return the minimum or maximum of a parametric expression calculated using values taken from a finite range. In this case the range is the set of dosages extracted from the historical storage ($stored), and the expression to calculate is the value itself ($d).

As a second example, we add a pre-condition to [INVOKE] alarm('high') stating that average time it takes the doctors to send back an acknowledgement must not exceed 200ms, and that a single invocation should never take more than 500ms.

```
let $range = retrieve(pID, null, null,
      '[INVOKE]alarm('high')/postcondition', $rt, 50);
(avg $t in $range; $t) < 200 && $rt < 500;
```

$rt is a special purpose keyword that can be used only in post-conditions and that refers to the amount of time it took the service to respond. $range, on the other hand, retrieves the last 50 $rts stored in [INVOKE]Alarm('high')'s post-condition. With respect to the previous example, we use the receive function to collect historical variables that belong to the entire process family. We are not looking at the response times that this process instance has experienced, but at all the response times experienced by all the instances of service TA.

As a third example, we are interested on predicating on the number of times a patient is hospitalised. This needs to be computed considering all the executions of the TA service for a given patient. We will state that hospitalisations should be less than 3. We could use this, for example, to signal that a fourth hospitalisation should not be requested without contacting the patient's doctor directly.

```
let $hosps = retrieve(pID, uID, null, null, $hospEvent, 10);
(num_Of $h in $hosps; $h) < 3;
```

$hosps contains the last 10 $hospEvent variables added to the historical storage. The num_Of aggregate function provided by the WSCoL language allows us to count how many values in a range satisfy a given property. In this case, we use this function to count how many $hospEvents (aliased as $h) were extracted from the historical storage, and compare its value with the constant 3.

To store these values in the historical storage we need to use the WSCoL store function. The following WSCoL code can be added throughout the TA service, in all those points in which a hospitalisation is performed.

```
let $hospEvent = 1; store $hospEvent;
```

Notice that in our retrieve function we only specify the user ID, and the process ID, but neither the instance ID nor the location in which the $hospEvents were stored. This is a must since there are different place in the process in which the hospitalisation may have been requested. We are interested in all of them.

## 4 ASTRO

In ASTRO [2,3], the approach to monitoring is characterised by three main features. (1) Monitors are independent software modules that run in parallel to BPEL processes, observe their behaviour by intercepting the input/output messages that are received/sent by the processes, and signal some misbehaviour or, more in general, some situation or event of interest. That is, the ASTRO approach does not require the monitored services to be decorated or instrumented to any extent, in order to guarantee direct usage of third party services. (2) The approach supports two different kinds of monitors: *instance monitors*, which observe the execution of a single instance of a BPEL process; and *class monitors*, which report aggregated information on all the instances of a given BPEL process. The latter assume a crucial importance to enact the computation of statistics and that therefore aggregate multiple executions of other processes. (3) Monitors are automatically generated and deployed starting from properties defined in RTML (Run-Time Monitor specification Language), which is based on events and combines them exploiting past-time temporal logics and statistical functionalities.

RTML stands on the notion of monitorable event, defining what can be immediately tracked by a monitoring process which observes the behaviour of services. Basic events correspond message exchanges (for instance, msg(TA.output = changeDose) denotes the emission of a recommendation of changing the insulin dose by the `TA` service), and creation and termination of service instances (denoted by the `start` and `end` keywords). Regarding *instance monitor* properties, RTML offers the ability to obtain monitor information of both logical and quantitative nature. The *logical* portion of RTML consists of standard boolean operators, and of a past-time linear temporal logic; that is, RTML allows for specifying properties on the whole past history of a given instance monitor, using "temporal" formulas such as $f_1$ `Since` $f_2$ (formula $f_1$ has been true since the last instant formula $f_2$ has been true) or `Once` $f$ ($f$ has been true at least once in the past). Such kinds of RTML expressions are useful to track down unexpected behaviours of a service, which in most cases can be represented by simple linear logic formulae. For instance, the following formula checks whether an hospitalisation is requested after at least one request to change the insuline dose:

```
msg(TA.output= sendAmbulance) && Once(msg(TA.output= changeDose))
```

The *numeric* portion of RTML allows counting events (operator `count`), and computing the time-span between events (operator `time`); this is very useful when checking, for instance, the QoS of the service instance being monitored. Indeed, the numeric and logical portions of RTML are closely integrated, so that it is possible to count the occurrences of complex behaviours (represented by temporal formulae), or vice versa, to trigger a boolean monitoring condition based on comparisons amongst numerical quantities (for instance, a certain event taking place more often than expected). An example of a numeric property is the fact that a hospitalisation is requested after three requests to change the insulin dose:

```
msg(TA.output = sendAmbulance) &&
  count(Once(msg(TA.output = changeDose))) = 3
```

An example of a time-based property is the fact that the time it takes the doctors to send back an acknowledgement (i.e., the time during which no `doctorEval` is received since the last `alarm ('high')` event) must never exceed 500ms:

```
time((!msg(TA.input = doctorEval)) Since
        (msg(TA.output = alarm('high'))) <= 500ms
```

Besides the instance monitors we considered so far, RTML also offers the possibility to specify *class monitors*, which aggregate monitoring results over all executions (or instances) of a given BPEL process. For instance, using an appropriate "class count" operator `Count`, it is possible to compute the total amount of times a certain property holds through all executions of a given process. For instance,

```
Count(msg(TA.output = sendAmbulance) &&
        count(Once(msg(TA.output = changeDose))) = 3)
```

computes the total number of times a hospitalisation has followed three recommendations to change the insulin dose. Similarly, an appropriate "class average" operator `Avg` allows us to compute the average times spent by services undertaking certain tasks. So,

```
Avg(time((!msg(TA.input = doctorEval)) Since
            msg(TA.output = alarm('high')))) <= 200ms
```

constrains the average time it takes to the doctors to answer to an emergency.

The ASTRO approach to support monitoring is based on converting RTML properties into state-transition systems that evolve on the basis of basic events of the process executions (e.g., reception and emission of messages). The states of the these state-transition systems codify the current evaluation of the formula, so that certain states are associated to the satisfaction of the monitoring requirement, while other states correspond to failures. A *monitoring engine* is responsible for receiving the relevant events from the BPEL execution engine, for correlating these events to the monitoring properties which depend on these events, for progressing the state of the state-transition systems that correspond to these properties, and for reporting failures and violations. This engine is built as an extension of ActiveBPEL [1], one of the most prominent engines for executing BPEL processes. The ASTRO extension allows for intercepting input/output messages and other relevant events such as the creation and termination of process instances. The extension also includes the ActiveBPEL admin console, which is exploited to report the information on the status of the monitors.

## 5 Comparison

In this section, we draw a comparison between Dynamo and ASTRO. This comparison will cover different aspects of a monitoring approach, namely: the kinds of basic events the approaches are able to monitor, the way these basic events can be combined in order to monitor more complex properties, the granularity of executions that a monitor can cover, and the level of integration of process execution and monitoring.

**Basic events.** We identify three different kind of basic events. *Messages*, i.e., the fact that a given message (with given values) is sent or received by the process; this kind of basic event can be managed by both approaches. *Control points*, i.e., the fact that the execution has reached a given point of the BPEL process; only Dynamo supports this kind of basic event. These events contain the variable values that are visible at that point in the process (in accordance with BPEL's own scoping rules). *Life cycle*, i.e., the possibility to monitor events related to the life cycle of a service execution, such as the fact that a new execution of a service is started, that the execution terminates with success or with an exception; only ASTRO supports this kind of basic events.

**Event combination.** We identify three different dimensions among which previous events can be combined in order to express complex monitoring properties. *Statistical* dimension, i.e., the possibility of producing aggregated information on a set of variables; both approaches support this dimension through operators such as `max`, `min`, `avg`, `sum`, `count`. *Time* dimension, i.e., the possibility of measuring durations and time intervals between events; this dimension is supported in a native way by ASTRO, while it can be encoded in Dynamo through external variables. *Temporal* (or behavioural) dimension, i.e., the possibility of expressing properties on the temporal evolution of the service, or behavioural patters that consist of sequences of events; this dimension is supported in a native way by ASTRO, through the use of temporal logic operators; it can be encoded in Dynamo through historical variables and quantifiers.

**Granularity.** This aspect is related to the granularity of process executions that are covered by a monitoring property. In particular, a property can refer to the following execution granularities. *Location*, i.e., the property is associated with a specific location of a BPEL process, and the monitoring is performed when the execution of a process instance reaches that location; Dynamo is based on this kind of granularity. *Process instance*, i.e., the property is associated with a single execution instance of a BPEL process, and the monitoring is performed through the instance execution; both approaches support this natively. *Process class*, i.e., the property is associated with all the executions of a BPEL process; ASTRO supports this kind of granularity in a native way (through class monitors); in Dynamo, these properties can be monitored using historical variables, quantifiers, and aggregated operators. *Cross-process*, i.e., the monitoring property can correlate events that refer to (execution instances of) different processes; ASTRO does not support this granularity; Dynamo requires encoding them in terms of "local" monitors and additional state variables.

**Integration level.** This aspect refers to the level of integration of the process execution engine with the monitor execution language. We identified the following levels of integration. *At the level of the BPEL specification*, i.e., the monitor is performed by instrumenting the BPEL specification with specific instructions that activate and advance the monitoring engine; this is the level of integration supported by Dynamo. *At the level of the BPEL engine*, i.e., the BPEL engine is tightly integrated with the monitoring engine and their executions are synchronised; neither Dynamo nor ASTRO support this level of integration. *Through asynchronous events* sent by the BPEL engine (or any other event source) to the monitoring engine; the monitoring engine is responsible for deciding which events are relevant for which monitors, and for advancing the monitoring task; ASTRO adopts this level of integration.

**Discussion.** A summary of the comparison of the Dynamo and ASTRO approaches is reported in Table 1. (The last row corresponds to the integrated approach defined in Section 6). The comparison shows that the two approaches are complementary under several aspects. In particular, Dynamo defines monitoring properties that are attached to specific locations of the BPEL specification, while ASTRO defines monitors that are associated to the whole execution of a BPEL specification. This difference also has effects on other aspects: on the level of integration between service execution and monitoring (strictly interconnected in the case of Dynamo, mediated by events in the case of ASTRO); on the capability to access the internal state of the BPEL process (supported by Dynamo but not by ASTRO); and on the capability of expressing properties that combine events in a complex way (easy to achieve in ASTRO, while slightly more difficult in Dynamo).

**Table 1.** Comparison of Dynamo and ASTRO approaches

| | Basic events | | | Combination | | | Granularity | | | | Integration level | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | messages | ctrl points | life cycle | statistical | time | temporal | location | instance | class | cross process | BPEL | engine | events |
| Dynamo | + | + | – | + | +/– | –/+ | + | + | –/+ | –/+ | + | – | – |
| ASTRO | + | – | + | + | + | + | – | + | + | – | – | – | + |
| Int.Appr. | + | + | – | + | + | + | + | + | + | + | + | + | + |

Legenda: +: yes –: no +/–: yes, with light encoding –/+: yes, with heavy encoding

## 6 Integration

In this section we discuss a possible way of integrating the approaches of Dynamo and ASTRO, trying to exploit as much as possible the complementarity of the two approaches and to achieve the highest level of expressiveness.

**Basic events.** We base our approach on WSCoL, i.e., we associate basic events to specific locations of the BPEL specification and we allow them to access the values of the internal BPEL variables. A basic event is hence defined by a *declaration*, a *location* within the BPEL code and by a *property*. The declaration defines the name, event parameters, and type of the event (see below). The location consists of an XPath expression and of a keyword defining the kind of triggering condition we want (a pre- or a post-condition), as in the Dynamo approach. The property is defined in an extension of the WSCoL language which extends the types of values properties can evaluate to. More precisely, we identify the following three types of WSCoL expressions. *Boolean* expressions, i.e., WSCoL expressions describing boolean conditions; all the WSCoL expressions in the Dynamo approach, hence including the examples reported in Section 3, are of this type. *Numeric* expressions, i.e., WSCoL expressions that evaluate to a numeric value; an example is the following expression for event `ratio(uId: string): numeric`, which computes the ratio between the current and the previous insulin dose:

```
let $doseNew = ($labResults/suggestedDose);
let $doseOld = retrieve(pID, uID, iID,
  '[INVOKE]changeDose/post-condition', 'doseStored', 1);
let $ratio = $doseNew / $doseOld;
$ratio;
```

*Tick* expressions, i.e., WSCoL expression that express the fact that a given event has occurred; these expressions are useful if a given event, associated to a given BPEL location has to be reported to higher level monitors only under certain conditions; an example is the following expression for event `ratioOutOfBounds(uId: string): tick`, which reports an event only if the insuline ratio is out of bounds:

```
let $doseNew=...; $let $doseOld=... ; $let ratio=...;
($ratio>0.95 && $ratio<1.05 ? NOTICK : TICK)
```

Notice that the `NOTICK` keyword can be used also with numeric or boolean expressions, in case the valued event has to be reported only under certain condition; an example is the following expression for event `ratioIfOutOfBounds(uId: string): numeric`, which reports the insulin ratio only if it is out of bounds:

```
let $doseNew=...; $let $doseOld=... ; $let ratio=...;
($ratio>0.95 && $ratio<1.05 ? NOTICK : $ratio)
```

**Composite monitor properties.** While basic events are based on Dynamo, the combination of these basic events into complex monitoring properties is based on the ASTRO approach. That is, we replace the basic events described in Section 4 with the events just introduced, while we keep the same operators and formulas defined in Section 4 for instance monitors and class monitors. More precisely, the syntax for events is "*name*`(%`*corr* `=` *par*`, ...)`", where *name* is the name of the event as defined in the WSCoL expression, *par* is the name of a parameter of the WSCoL declaration, and *corr* is a correlation variable, which is used to correlate events of different processes in a class monitor. Assume for instance that two basic events have been defined for service `TA`, namely `ratio (uId: string): numeric`, and `hospitalisation(uId: string): tick`. Then, the following *instance* monitor checks whether hospitalisation has been decided after the insulin dose has been incremented three times:

```
hospitalisation & count(Once(ratio > 1)) >= 3
```

The following *class* monitor reports the number of cases in which such an increment in the dose has lead to a hospitalisation for a given patient:

```
Count(hospitalisation(%pat = uId) &
        count(Once(ratio(%pat = uId) > 1)) >= 3)
```

Notice the usage of the correlation variable to select only the service executions corresponding to the same patient. If the correlation variable is removed, then a total count for all patients is computed:

```
Count(hospitalisation & count(Once(ratio > 1)) >= 3
```

We remark that the explicit correlation mechanism we adopt allows for the definition of cross-process class monitors. Indeed, it is easy to correlate events defined in different processes. Assume for instance that a different service monitors blood pression and defines basic events `lowPression/normPression(uId: string): TICK`. If we want to monitor the case where an increase in the insulin rate is recommended after a low blood pression is reported, we can define the following monitor:

```
(ratio(\%pat = uId) > 1) &
   ((!normPression(\%pat = uId)) Since (lowPression(\%pat = uId)))
```

**Architecture.** Since the monitoring language we have defined combines the localised basic events *à la* Dynamo with instance and class monitors approach *à la* ASTRO, the monitoring architecture is also a combination of Dynamo and ASTRO. More specifically, the computation of basic events is achieved by instrumenting the BPEL processes. This requires a slight modification of the code that is weaved by Dynamo into the execution engine. The instrumented processes, which are then deployed and executed on a standard BPEL engine, send information on the occurrence and value of basic events to the monitoring engine. This communication may occur through both asynchronous events and synchronous communications, allowing for both asynchronous and synchronous work by part of the execution and monitoring engine. The monitoring engine performs a correlation step in order to decide which monitor instances are relevant for a given basic event (and possibly instantiates new monitors if necessary). The monitoring engine also updates the status of the relevant monitor instances and reports problems and violations. The generation of the run-time components for the monitors combines the Dynamo and ASTRO approaches as well. Indeed, the former approach is exploited for instrumenting the BPEL processes, while the latter approach is responsible of generating the monitor instances executed by the monitoring engine.

## 7 Conclusions

In this paper, we proposed a novel monitor approaches that leverages Dynamo and ASTRO, two existing approaches for the run-time monitoring of BPEL orchestrations which have been developed within the research groups of the authors of this paper. The new approach is able to monitor a wider range of features than any other monitoring approach for BPEL orchestration the authors are aware of.

Our future work will concentrate on the implementation of the proposed integrated approach and on its thorough evaluation on real-world applications; this will also require detailing the definition of the novel language and of the architecture described in Section 6. On the side, we are also interested in evaluating the possibility to integrate other approaches that were not developed by the authors. In a longer term, we plan to investigate extensions of the proposed approach that include reactions to the anomalies and emergencies identified by the monitoring; the goal is to close the loop and influence the behaviour of the services according to the information collected during the monitoring. We also plan to extend the monitoring approach to the case of compositions of web services that are distributed among different nodes of the network; in this case, both the monitoring language and the monitoring architecture will have to deal with the necessity of collecting events over the network.

## References

1. ActiveBPEL. The Open Source BPEL Engine, `http://www.activebpel.org`
2. Barbon, F., Traverso, P., Pistore, M., Trainotti, M.: Run-Time Monitoring of the Execution of Plans for Web Service Composition. In: Proc. ICAPS 2006, pp. 346–349 (2006)
3. Barbon, F., Traverso, P., Pistore, M., Trainotti, M.: Run-Time Monitoring of Instances and Classes of Web Service Compositions. In: Proc. ICWS 2006, pp. 63–71 (2006)
4. Baresi, L., Ghezzi, C., Guinea, S.: Smart monitors for composed services. In: Proc. ICSOC 2004, pp. 193–202 (2004)
5. 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)
6. Baresi, L., Guinea, S.: A dynamic and reactive approach to the supervision of BPEL processes. In: Proc. ISEC 2008, pp. 39–48 (2008)
7. Beeri, C., Eyal, A., Milo, T., Pilberg, A.: Monitoring business processes with queries. In: Proc. VLDB 2007, pp. 603–614 (2007)
8. Bianculli, D., Ghezzi, C.: Monitoring Conversational Web Services. In: Proc. IW-SOSWE 2007 (2007)
9. Mahbub, K., Spanoudakis, G.: Run-time Monitoring of Requirements for Systems Composed of Web Services: Initial Implementation and Evaluation Experience. In: Proc. ICWS 2005, pp. 257–265 (2005)
10. Momm, C., Malec, R., Abeck, S.: Towards a Model-driven Development of Monitored Processes, Wirtschaftsinformatik, vol. 2 (2007)
11. Roth, H., Schiefer, J., Schatten, A.: Probing and monitoring of WSBPEL processes with web services. In: Proc. CEC-EEE 2006 (2006)

# Towards Goal-Driven Self Optimisation of Service Based Applications

Andreas Gehlert and André Heuer

University of Duisburg-Essen, Schützenbahn 70
45117 Essen
{andreas.gehlert,andre.heuer}@sse-uni-due.de

**Abstract.** Service based applications are constructed to be easily adaptable to changing environments. This adaptation was primarily investigated with respect to monitoring events, e. g., a service based application is adapted when the execution of a service fails. In this paper we focus on the adaptation of service based applications due to newly available service. In this respect we discuss whether a service of the service based application should be replaced by a service, which becomes available. From the requirements engineering perspective we argue that a service based application may be adapted when the new services contribute better to the goals of the service based application. In addition, we show that it may also be valuable to adapt a service based application when newly available services provide more functionality than the ones previously used. Both analyses are based on model comparison techniques with Tropos goal models and Tropos' reasoning techniques.

**Keywords:** Adaptation of Service Based Applications, Model Comparison, Goal Models, Requirements Engineering.

## 1 Introduction

In traditional software engineering three different motivations for software maintenance and corresponding software adaptations are distinguished [1, p. 493]: Corrective maintenance aims to eliminate errors in the software which were not discovered during the testing phase. Adaptive maintenance reacts to changes in the system's context and adapts the software to new requirements. Perfective maintenance aims to improve the current software, e. g. its performance.

Due to an increased number of mergers in the industry [2, pp. 12] and an increased pressure on IT departments to improve their efficiency [3, pp. 5] service based applications (SBA) evolved, which particularly facilitate adaptation. This adaptation is enabled by defining technologies for describing, finding and composing services. By *service* we understand any computational resource offered over a network [4, p. 51].

One current vision is to engineer SBAs to enable self-optimization. Such self-optimising systems "… continually seek ways to improve their operation, identifying and seizing opportunities to make themselves more efficient …." [5, p. 43] Although

© Springer-Verlag Berlin Heidelberg 2008

current adaptation and monitoring approaches aim to eliminate errors in the SBA (corrective maintenance) or to improve the performance of the SBA (perfective maintenance), they are usually technology-centred and disregard requirements engineering (RE) aspect of SBAs.

In this paper we complement current adaptation and monitoring approaches with a RE perspective. Our approach strives to achieve self-optimising SBAs (perfective maintenance) by means of fulfilling given requirements. To focus the paper we limit the discussion of adaptation scenarios to the analysis whether existing services in the SBA should be replaced by newly available services. Other adaptation techniques, e. g. the reconfiguration of the workflow or an adaptation of the service infrastructure are not addressed.

If a new service is available, the requirement engineer investigates whether the new service improves the SBA with respect to its requirements. In order to use a new service in a SBA, we raise two requirements: First, the service must "fit" in the existing SBA. Second, the service must contribute to the SBA's requirements better than previously available services. The first requirement ensures that the service provides the functionality needed by the SBA. The second requirement ensures that the new service is superior to existing services.

To satisfy both requirements, we chose a goal-driven approach. The main reason for this choice is the availability of reasoning techniques for goal models, which allows analysing the effect of the satisfaction of a single goal on the whole goal model. This facilitates the analysis of the impact of a single service on the entire SBA. To address requirement one, we compare the goal model of the SBA and the goal model of the new service. The service is only useable if its goal model is identical or similar to the goal model of the SBA. To find out whether the service provides higher satisfaction ratios for the SBA's goal model, we use the goal reasoning mechanisms provided in [6]. We argue, that an adaptation of an initial SBA is advisable if all goals are at least as satisfied as in the initial situation but one goal has a higher satisfaction (pareto principle). In case that some goals of the SBA achieve higher satisfaction rates in the new situation and some goals achieve lower satisfaction rates, the requirements engineer may decide on the adaption of the SBA. Lastly, an adaptation may also be advisable when the new service provides additional functionally.

We chose to use Tropos as goal modelling approach [7, 8]. The rationale for using Tropos is threefold: First, Tropos is a comprehensive approach to develop software systems from early requirements to its implementation. We are particularly interested in the early RE activities, which are well covered by Tropos. Second, Tropos was already applied to the service discipline, e. g., it was already shown that it is applicable to SBAs [e. g. 9, 10-13]. Third, Tropos comes with a formalisation which allows analysing the influence of the satisfaction of one goal on the entire goal model [6].

The paper is organised as follows: In section 2 we introduce Tropos' goal modelling techniques, which are used throughout this paper. In section 3 we show how goal models can be used to decide whether a new service should be used in an existing SBA, e. g. whether an adaption of this SBA is beneficial. We discuss the limitations of our approach along its assumptions in section 4, review the related work in section 5 and provide conclusions in section 6.

## 2 Goal Modelling in Tropos

Tropos rests on the agent oriented paradigm and uses goal modelling techniques known from i* [14] for analysing early and late requirements. These early requirements are documented as actor and goal models. *Actor models* include actors, their goals and their dependencies. The actor diagram is complemented by a *goal model* for each actor. This goal model shows the decomposition of the actor's goals into sub-goals and plans (tasks in i*; cf. Fig. 1 for an example).

Since we are only interested in the SBA (represented as actor in Tropos) and not its interrelation with other actors, we only use Tropos' s goal models. Its main concepts are actors, goals, resources and plans. These elements are connected with decomposition, contribution and means-end links. An *actor* represents an organisational unit, a position or a role. The actor's strategic interests are represented by *goals*. Goals are further divided into hard-goals and soft-goals. While *hard-goals* have clear cut measurement criteria to specify its fulfilment, *soft-goals* do not have such criteria. In this paper we use hard-goals to model functional and soft-goals to model quality requirements. Goals can be decomposed using And/Or *decomposition* links. A *plan* is an activity which may contribute to a goal. A plan may also *contribute* positively or negatively to soft-goals. *Means-end* links are used to represent which plan or goal (means) is used to fulfil a goal (end, [7, pp. 206]).

In a SBA each plan describes a service, which realises this plan [9, p. 21]. To implement a SBA, it is, therefore, necessary to find services, whose descriptions fit the plan. Consequently, a SBA in the Tropos early RE perspective is a set of services fitting a set of plans. The description of this fitness relation is described in the next section.

## 3 Using Goal-Models for Adapting SBAs

To define the fitness relation between a plan and a service we need to define what a plan and a service is. A plan in Tropos is defined by its name and its relations to goals and soft-goals via a set of means-end links $M$ and via a set of contribution links $C$. Since we later use a satisfaction propagation mechanism to describe the dependencies between different goals, it is sufficient to include only directly connected means-end and contribution links in the definition of a plan. Thus, we can define a plan as a tuple of sets of means-end and contribution links: $plan = < M, C >$.

To compare plans and services we assume:

(A1) Services are described by Tropos goal models. These goal models are registered together with the service in a service registry.

(A2) Each service's goal model contains only the name of the service as plan and only goals and soft-goals connected by means-end and contribution links. Consequently, the service goal model is structurally equivalent to the plan's goal model.

Assumption (A1) is critical insofar as service providers need to describe their services with goal models. A detailed discussion of this assumption is postponed to section 4. Assumption (A2) is not critical as we describe below an algorithm, which produces this sub-model from any Tropos goal model.

According to assumption (A2) services and plans are described by the same elements. Consequently a service is also a tuple of sets of means end and contribution links: $service = < M, C >$. A means-end link $m \in M$ is a connection between one Tropos model element $e$ and a goal $g$. Each contribution link $c \in C$ is a connection between one Tropos model element $e$ and a soft-goal $s$. It is attributed with a quantitative number to express the strength $\omega$ of the contribution to this soft-goal: $c = < e, s, \omega >$.

As we concentrate on requirements engineering for adapting SBAs, we assume that a SBA is already running and that its initial set of requirements are expressed as Tropos goal model. In addition, each service in this SBA fits to one plan in the requirements specification.

After the initial SBA is operating, the service provision is monitored, e. g. by regularly querying service registries. When new services are available, an adaptation cycle is triggered. The first activity in this cycle extracts a goal model for each plan of the SBA's goal model. These goal models are in turn compared to the service's goal model. After this comparison, the goal achievements for all goals in the Tropos goal model are calculated based on formal reasoning techniques described below. These results are than used to decide about the adaptation of the SBA. After this adaption the process starts again. The underlying assumption of this RE process can be formulated as follows:

(A3) New services become available over time.

This assumption is fair because the flexibility of SBAs is only feasible when new services are made available over time.

In the next step we need to describe how the goal model for each plan can be extracted from the entire Tropos model. Extracting a goal model for each plan is necessary since we want to delegate the execution of each plan to a service and since the Tropos model represents the entire SBA and not only one individual service. Intuitively the goal model for each plan contains all its contribution and means-end links and all the connected elements. Both link types are important because we want to know how the service for each plan influences the goal achievement of the entire SBA. In sum, the plan's goal models contains: the plan, all connected goals via contributions links and all connected goals via means-end links.

To illustrate the extraction of the goal model for one plan, Fig. 1 provides an example of a retailer system [partially taken from 9, p. 22]. The main goal of this retailer system is to sell products. A goal analysis revealed the goals "order handling" and "cataloguing" (And decomposition). Tasks and soft-goals were added to the diagram and were connected to these goals with means-end relationships and contribution links respectively. The warehouse service for instance contributes positively with the strength of *+0.3* to the soft-goal "performance". For the initial SBA, each plan is the description of a service. At the beginning of the RE process, the goal model of this service is identical to the extracted goal model of the plan. The goal model for the plan "eShop Service" for instance contains the hard-goal "Cataloguing" and the soft-goals "Performance", "Availability", "Transaction Reliability" as well as the respective means-end and contribution links (Fig. 1, right).

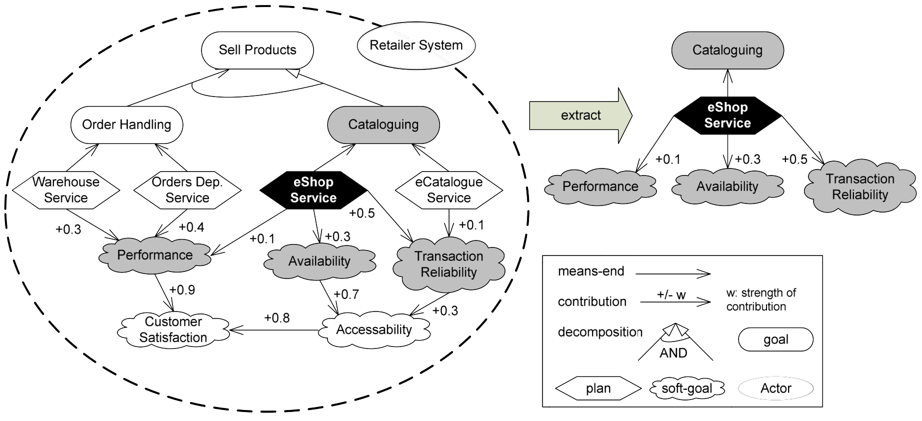


**Fig. 1.** Extracting the Sub-Model for the Plan "eCatalogue Service" [example partially taken from 9, p. 22]

## 3.1 Comparing the Service and the Plan Goal Models

After the specification of the initial requirements and the extraction of the goal model for each plan, we need to monitor the service provision. If the service provision changed we want to find out whether the newly available services "fit" the existing plans better than the initial set of services. In this section we define this fitness relation. This fitness relation is based on the comparison of the plan's goal model and the service's goal model in accordance with assumptions (A1) and (A2).

The systematic analysis of model comparison conflicts was initially developed in the data modelling discipline. Batini et al. distinguish between naming conflicts, structural conflicts and type conflicts.

- *Naming conflicts* arise due to the different usage of the natural language in models [14, p. 344].
- *Type conflicts* can be traced back to the divergent usage of the modelling language, e. g. to express one and the same real world phenomenon as entity type or as attribute [14, p. 346].
- *Structural conflicts* arise when a real world proportion is differently reconstructed by different modellers, e. g. because of different goals of the modelling project [14, p. 346].

A model comparison technique aims to identify the before-mentioned conflicts. *Naming conflicts* can be resolved by analysing the homonyms and synonyms used in the models and by renaming the respective model elements so that identical names in both models have the same and different names in the models have different meanings. As this problem was already solved previously [for an overview cf. 15, p. 344], we assume here that naming conflicts were already resolved:

(A4) The service's goal model and the plan's goal model use a shared ontology, i.e. two goals with the same name are identical and two goals with different names are different.

Resolving *type conflicts* means to define a similarity relation between equivalent or similar model structures. Our models contain only hard-goals, soft-goals, contribution links and means-end links (cf. assumption (A2)). In addition, hard-goals describe functional requirements and soft-goals represent quality requirements. As functional requirements in the service domain are described by the web service description language (WSDL) and non-functional requirements are described by service level agreements (SLA) it follows that hard-goals and soft-goals are mutually exclusive and cannot be resolved in the type conflict analysis. The remaining elements are means-end and contribution links. Means-end links are used whenever plans and soft-goals provide a means to achieve a goal [7, p. 208]. Consequently, the means fully satisfies the goal or soft-goal, which is identical to a contribution link with a degree of *+1.0*.

To define the fitness relation between a plan and a service we introduce the functions $name()$, which returns the name of a goal model element and $type()$, which returns the type of a model element. Based on this analysis of type conflicts we can now define when a plan matches a service description:

$$
\begin{aligned}
& p \xrightarrow{fits} s \Leftrightarrow \\
& \forall m_P \in M_P : \left( \begin{array}{l} \exists m_S \in M_S : \big(name(g_P) = name(g_S) \wedge (type(g_P) = type(g_S)\big) \vee \\ \exists c_S \in M_S : \big(name(g_P) = name(s_S) \wedge type(g_P) = soft\text{-}goal \wedge \omega_s = 1\big) \end{array} \right) \wedge \\
& \forall c_p \in C_P : \left( \begin{array}{l} \big(\exists c_S \in C_S : name(s_P) = name(s_S)\big) \vee \\ \big(\exists m_S \in M_S : \big(name(s_P) = name(g_S) \wedge type(g_S) = soft\text{-}goal\big)\big) \end{array} \right)
\end{aligned}
$$

This fitness relation holds when:

1. Each means-end link $m_P$ in the plan's goal model exists also in the service's goal model ( $m_S$ ) and the connected goals have identical names and types. As a means end link can also be represented as contribution link, it follows: Each means end link $m_P$ of the plan's goal model exists in the service's goal model as contribution link with the strength $\omega_s = 1$ and the connected goals have the same name and the plan's goal is a soft-goal.
2. It must additionally hold that for each contribution link in the plan's goal model $c_p$ there is either a contribution link $c_S$ in the services goal model and the connected goals have identical names. Alternatively, the contribution link $c_p$ may also be represented as means end link $m_S$ in the service's goal model. In this case both connected goals must have the same name and the service's goal must be a soft-goal.

Lastly, *structural conflicts* cannot be resolved and represent the real differences of the models. However, they can for instance be used to analyse whether a goal model 1 includes goal model 2 but has additional hard- and/or soft-goals.

## 3.2 Decision Support for Adapting a SBA

An adaptation of the SBA is only feasible if the relation $p \xrightarrow{fits} s$ holds – otherwise the service does not provide the required functionality needed by the SBA. When a new service is registered in a registry (assumption (A3)) and the fitness relation holds for this service for one plan, we can distinguish four situations:

Situation 1 *Equal Goal Satisfaction*: The goal model of the new service is identical to the plan's goal model. In this case the new service can be used as substitute of the existing service, e. g. when the existing service fails.

Situation 2 *Different Goal Satisfaction*: The new service may contribute differently to existing soft-goals by assigning different strengths to contribution links. These different strengths are further propagated in the goal model and may lead to different satisfaction ratios of goals and soft-goals. The adaptation decision is based on these new goal satisfaction ratios.

Situation 3 *Goal extension*: The new service may provide additional functionality not used in the initial SBA. This new functionality is expressed as additional hard-goals in the service goal model, which do not correspond to any goal in the plan's goal model (structural conflict). The SBA may be adapted accordingly to exploit the additional functionality of the new service.

Situation 4 *Goal reduction*: The new service may provide less functionality in comparison to the one used in the current SBA. The requirements engineer may decide using this service in combination with another service, which together fulfil the requirements of the SBA better than the services used previously.

In the following we demonstrate the calculation of the goal satisfaction values in accordance to the newly available service. We use the quantitative reasoning techniques in Tropos goal models presented in [6, p. 10]. The algorithm presupposes that each goal has two variables $Sat(G)$ and $Den(G)$ describing the satisfiability and deniability of the goal. These variables are computed according to the strength $\omega$, which is annotated to contribution links. This strength describes the impact of one goal on another goal. Due to space limitations, we restrict ourselves to goal satisfyability.

For each contribution link $G_2 \xrightarrow{\omega+s} G_1$ with the strength $\omega$ Giorgini et al. define the following propagation axiom: $G_2 \xrightarrow{\omega+s} G_1 : Sat(G_2) \geq x \rightarrow Sat(G_1) \geq (x \otimes \omega)$ [6, p. 12]. The operator $\otimes$ is defined as $p_1 \otimes p_2 =_{def} p_1 \cdot p_2$. In addition, we assume that means-end links can be treated like contribution links with $\omega = 1$. We can now use the axiom to calculate $Sat(G)$ for each goal of the goal model. This goal propagation assumes the following:

(A5) The strengths $\omega$ of all service's goal models are comparable, e. g. they are measured objectively.

This assumption is necessary to actually compare the satisfaction ratios of the soft-goals among different services. We discuss this assumption in section 4.

The result of the label propagation can be presented as bar chart. The y-axis is labelled with goals and soft-goals, the x-axis is labelled with the degree of satisfaction and the bars show the degree of satisfaction of the different goals. The bar chart representation of the goal model in Fig. 1 is depicted in Fig. 2 (black bars).

Assume that new services were registered in a service registry. To use these new services we require, that their goal models are structurally identical to the plan's goal model. Two goal models of new services, which fulfil this requirement are depicted in Fig. 2 (gray bars). Both models conform to the before-mentioned situation 2.

In comparison to the goal model in Fig. 1, the contribution links of service ❶ in Fig. 2 to Transaction Reliability and to Availability have an increased strength (*0.6* instead of *0.5* and *1.0* instead of *0.3*). Using service ❶ means to achieve higher fulfilment rates for all soft-goals. Consequently, using service ❶ is beneficial from the RE perspective and should therefore be used.

Service ❷ in Fig. 2 has a reduced strength for the contribution link to Reliability (*0.3* instead of *0.5*) but an increased strength for the contribution link to Availabilty (*0.*7 instead of 0.*3*). The goals Accessibility, Availability and Customer Satisfaction have now a higher satisfiability. However, the satisfiability of the goal Reliability dropped. A RE expert has to decide about the adaptation of the SBA. In this case this may be valuable because the goal Consumer Satisfaction increased slightly. The RE expert has to balance this advantage with the disadvantage of a lower satisfiability of the goal Reliability.

## 4 Discussion

The results of this paper are limited to the assumptions (A1) – (A5). In particular it relies on the assumption that new services become available over time (A3). This assumption is fair because changing environments lead to new IT solutions in the past and will most likely lead to new services in the future.

Assumption (A2) is based on assumption (A1) and requires that each provider provides a goal model as description of his/her service (assumption (A1)) and that this goal model consists only of one plan representing the service as well as goals and soft-goals directly connected to the plan with means-end and contribution links. This assumption is also less critical since we explained how such a reduced goal model can be extracted from a larger goal model.

Assumption (A4) requires that service providers and service consumers use a shared vocabulary. Although this assumption is not realistic, it can be eliminated by linguistic approaches, which resolve homonyms and synonyms. For instance, WordNet [16] was successfully used to resolve homonyms and synonyms in the SeCSE approach to requirements engineering [17]. In other words, assumption (A4) was helpful to focus this paper but can be overcome by using existing approaches.

The most critical assumptions are (A1) and (A5). They require that service providers provide a goal model for each service (assumption (A1)) and that the strengths of the contribution links are objectively comparable between different services. Both assumptions seem unrealistic. However, instead of forcing service providers to describe their services with goal models, these goal models may be generated. The central plan element can be generated according to the service's name. The functional requirements of this service are described in a WSDL document. Consequently, the hard-goals are represented by the methods contained in this WSDL document. In

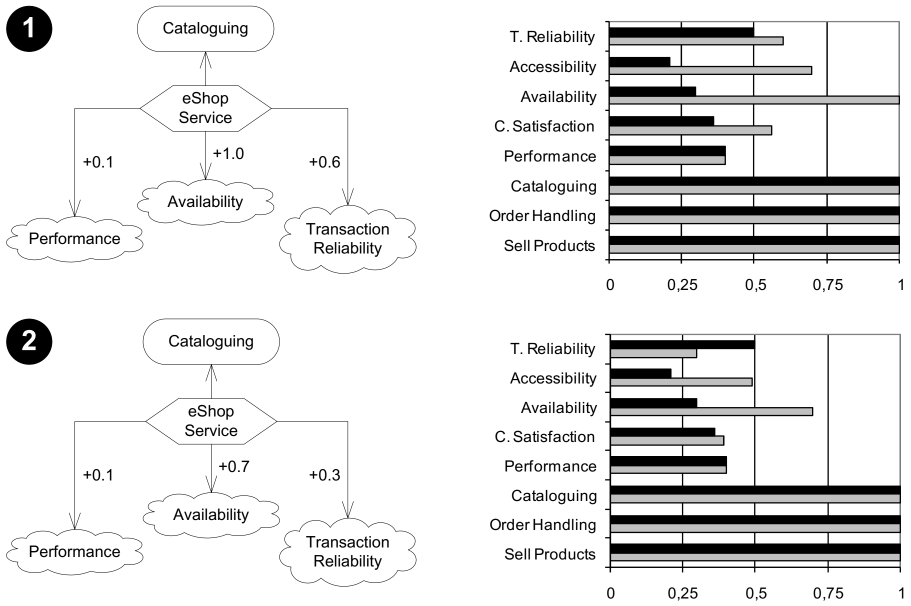


**Fig. 2.** Quantitative Comparison of Goal Models

addition, a SLA describes the quality requirements for a service and it may be used to generate soft-goals. If the quality characteristics are quantified in the SLA, this quantification can also be used to calculate the strengths of the contribution links. Assume that the requirement for the parameter “response time” is 1s. Service 1 has a response time of *2s* and service 2 a response time of *3s*. Consequently, the strengths of the contribution links are *0.5* for service 1 and *0.33* for service 2. The feasibility of this approach, however, is subject to further research.

## 5 Related Work

Although Tropos was applied in the service domain, these applications do not explain when to adapt a SBA. Aiello and Giorgini for instance explore quality of service aspects using Tropos actor models [9]. The authors use Tropos' formal reasoning techniques in [6] to calculate the fulfilment of a goal structure according to a given set of services. As the approach by Aiello and Giorgini does not cover the adaptation of a SBA, our approach is an extension to [9]. In another approach Penserini et al. explore how Tropos can be used to develop SBAs. However, the authors do not focus on adaptation. Another application of Tropos was put forward by Pistore et al. in [13]. The authors explain how SBAs can be developed by step-wise refining plans and complementing these plans with a formal workflow definition. Since the focus of Pistore et al. is on deriving service compositions, the authors do not cover adaptation issues.

A similar approach to ours was put forward by Herold et al. in [18]. The authors relate existing components to goal models. This relation is established by so called generic architectural drivers. These drivers enable the selection of existing components, which fit with the goals and soft-goals of the goal model. Herold et al.'s approach focus on finding appropriate components and refining the initial goal model with the help of these components. However, the approach does not address adaptation.

Another RE approach, which is similar to ours, was put forward in the SeCSE project [19]. In SeCSE initial requirements are formulated as goal models [19, pp. 21] or use cases [17, 20-22], which are than translated into services queries [19, p. 31]. These services queries are sent to a registry. The resulting services are used to refine the initial set of requirements. However, in SeCSE the focus was on refining requirements according to the current service provision but not on adapting existing SBAs.

## 6 Conclusion

In this paper we presented an approach towards self-optimisation of SBAs. In particular we showed how to decide whether a SBA should use newly available services. Informally a SBA should use a new service if this service has the same functionality as the existing service but fulfils the requirements of the SBA better, e. g. has a higher performance than the existing service.

To measure the fulfilment of the requirements we proposed to use Tropos and its formal reasoning techniques. The starting point of our approach is a goal model of the SBA's requirements. Each plan in this goal model is the description of a service. When a new service is available, the Tropos goal model is compared to the goal model of this service. The new service is superior to the previous one, if the goal satisfaction ratios for all goals in the SBA requirements specification are higher compared to the old service. In addition, the requirements engineer may also decide to adapt the SBA, when the new service is superior with respect to some goals but inferior to others or when the new service provides additional functionality expressed as additional goals.

To analyse the impact of one particular service on the whole SBA we use Tropos formal reasoning techniques. These reasoning techniques allow to propagate the local satisfaction ratios of goals of one service to the whole goal model and, thus, to analyse the dependencies of the different services within this SBA.

Our approach clearly shows that the prerequisite of the adaptation of SBAs discussed here requires that service providers and service consumers speak the same language, e. g. that both parties agree on a shared ontology. In addition, the difficulties of measuring quality aspects of services are even more evident since a comparison between two services with respect to their quality attributes requires a shared metrics between service providers and service consumers. One approach, which tackles this problem can be found in [23].

Our approach can be extended in four ways: First, a revised version may also contain the notion of goal deniability introduced in [6], which we left out due to space limitations. Second, we need to show that the proposed approach to generate service goal models from WSDL and SLA specification is feasible. This helps to overcome the restrictive assumption underlying our approach. Third, the combination of the proposed requirements engineering approach with adaptation techniques – in particular with self optimisation techniques – would provide the missing link to the service engineering domain. Forth, the approach should be formally and empirically validated to prove its efficiency.

## 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. Swanson, E.B.: The Dimensions of Maintenance. In: International Conference on Software Engineering (ICSE 1976), San Francisco, CA, USA, pp. 492–497 (1976)
2. Budzinski, O., Kerber, W.: Megafusionen, Wettbewerb und Globalisierung: Praxis und Perspektiven der Wettbewerbspolitik, vol. 5. Lucius & Lucius, Stuttgart (2003)
3. Mutschler, B.: Die IT und betriebliche Informationssysteme im Spannungsfeld von Innovation und Wirtschaftlichkeit. Ulm (2006)
4. Colombo, M., Nitto, E.D., Penta, M.D., Distante, D., Zuccalà, M.: Speaking a Common Language: A Conceptual Model for Describing Service-Oriented Systems. In: Benatallah, B., Casati, F., Traverso, P. (eds.) ICSOC 2005. LNCS, vol. 3826, pp. 48–60. Springer, Heidelberg (2005)
5. Kephart, J.O., Chess, D.M.: The Vision of Autonomic Computing. IEEE Computer 36, 41–50 (2003)
6. Giorgini, P., Mylopoulos, J., Nicchiarelli, E., Sebastiani, R.: Formal Reasoning Techniques for Goal Models. Journal on Data Semantics, 1–20 (2003)
7. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Tropos: An Agent-Oriented Software Development Methodology. Autonomous Agents and Multi-Agent Systems 8, 203–236 (2004)
8. Castro, J., Kolp, M., Mylopoulos, J.: Towards Requirements-Driven Information Systems Engineering: The Tropos Project. Information Systems 27, 365–389 (2002)
9. Aiello, M., Giorgini, P.: Applying the Tropos Methodology for Analysing Web Services Requirements and Reasoning about Qualities. UPGRADE: The European Journal for the Informatics Professional 5, 20–26 (2004)
10. Lau, D., Mylopoulos, J.: Designing Web Services with Tropos. In: International Conference on Web Services (ICWS 2004), San Diego, CA, USA, pp. 306–313 (2004)
11. Misra, S.C., Misra, S., Woungang, I., Mahanti, P.: Using Tropos to Model Quality of Service for Designing Distributed Systems. In: International Conference on Advanced Communication Technology (ICACT 2006), Phoenix Park, Gangwon-Do, Republic of Korea, pp. 541–546 (2006)
12. Penserini, L., Perini, A., Susi, A., Mylopoulos, J.: From Stakeholder Needs to Service Requirements. In: 2nd International Workshop on Service-Oriented Computing: Consequences for Engineering Requirements (SOCCER 2006), Minneapolis, Minnesota, USA, pp. 8–17 (2006)
13. Pistore, M., Roveri, M., Busetta, P.: Requirements-Driven Verification of Web Services. Electronic Notes in Theoretical Computer Science 105, 95–108 (2004)
14. Yu, E.: An Organisational Modelling Framework for Multiperspective Information System Design. Requirements Engineering 1993 – Selected Papers. Department of Computer Science, University of Toronto, Toronto, pp. 66–86 (1993)
15. Rahm, E., Bernstein, P.A.: A Survey of Approaches to Automatic Schema Matching. VLDB Journal 10, 334–350 (2001)

16. Miller, G.A.: WordNet - Princeton University Cognitive Science Laboratory (2006), `http://wordnet.princeton.edu/`
17. Jones, S.V., Maiden, N.A.M., Zachos, K., Zhu, X.: How Serivce-Centric Systems Change the Requirements Process. In: 11th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ 2005), Porto, Portugal, pp. 105–119 (2005)
18. Herold, S., Metzger, A., Rausch, A., Stallbaum, H.: Towards Bridging the Gap between Goal-Oriented Requirements Engineering and Compositional Architecture Development. In: 2nd Workshop on Sharing and Reusing architectural Knowledge Architecture, Rationale, and Design Intent (SHARK-ADI 2007), Minneapolis, USA (2007)
19. Maiden, N.: Service Centric System Engineering: A2.D5 SeCSE Requirements Process V2.0. City University London (2006)
20. Zachos, K., Maiden, N.: Web Services to Improve Requirements Specifications: Does It Help? In: Paech, B., Rolland, C. (eds.) REFSQ 2008. LNCS, vol. 5025, pp. 168–182. Springer, Heidelberg (2008)
21. Zachos, K., Maiden, N.A.M., Zhu, X., Jones, S.: Discovering Web Services to Specify More Complete System Requirements. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) CAiSE 2007 and WES 2007. LNCS, vol. 4495, pp. 142–157. Springer, Heidelberg (2007)
22. Zachos, K., Zhu, X., Maiden, N., Jones, S.: Seamlessly Integrating Service Discovery Into UML Requirements Processes. In: Proceedings of the 2006 International Workshop on Service-Oriented Software Engineering (SOSE 2006), Shanghai, China, pp. 60–66 (2006)
23. Zachos, K., Dobson, G., Sawyer, P.: Ontology-aided Translation in the Comparison of Candidate Service Quality. In: Proceedings of the 4th SOCCER Workshop, Barcelona, Spain, September 8 (2008)

# Towards Correctness Assurance in Adaptive Service-Based Applications*

Raman Kazhamiakin[1], Andreas Metzger[2], and Marco Pistore[1]

[1] FBK-Irst, via Sommarive 18, 38050, Trento, Italy
{raman,pistore}@fbk.eu
[2] SSE, University of Duisburg-Essen, Schützenbahn 70, 45117 Essen, Germany
andreas.metzger@sse.uni-due.de

**Abstract.** Service-based applications (SBAs) increasingly have to become adaptive in order to operate and evolve in highly dynamic environments. Research on SBAs thus has already produced a range of adaptation techniques and strategies. However, adaptive SBAs are prone to specific failures that would not occur in "static" applications. Examples are faulty adaptation behaviours due to changes not anticipated during design-time, or conflicting adaptations due to concurrently occurring events. For adaptive SBAs to become reliable and thus applicable in practice, novel techniques that ensure the correctness of adaptations are needed. To pave the way towards those novel techniques, this paper identifies different kinds of adaptation-specific failures. Based on a classification of existing adaptation approaches and generic correctness assurance techniques, we discuss how adaptation-specific failures can be addressed and where new advanced techniques for correctness assurance of adaptations are required.

## 1 Introduction

A wide range of research approaches addresses the problem of adaptation in Service-Based Applications (SBAs). Those include solutions for the definition and realization of adaptation requirements as well as strategies for adapting the applications to new situations and to react to various events and failures. As SBAs increasingly have to operate and evolve in highly dynamic environments and must dynamically and autonomously accommodate for various changes and events, dynamic adaptation is particularly important since it allows for automated and timely modification of the underlying application.

In general, the approach to dynamic SBA adaptation may be described as follows: At design time, the service integrator prescribes the *adaptation specification*, by $(i)$ identifying the dynamic part of the application (i.e., what can change or can happen in the application or its environment that requires the application to adapt), $(ii)$ defining the adaptation requirements that the underlying application should respect, and $(iii)$ selecting the strategies to realize these requirements. During run-time, the application detects the changes and executes the selected adaptation strategies according to the adaptation specification.

* 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).

© Springer-Verlag Berlin Heidelberg 2008

While the functional behavior of an SBA typically is controlled by user input, dynamic adaptation is triggered by additional information about the application and its context (e.g., failures of constituent services or different network connectivity). As a consequence, dynamic adaptation may lead to new kinds of failures. Those adaptation-specific failures include:

– the execution of the adaptation specification may fail, when the application encounters a situation that is not covered by the adaptation specification, i.e. which has not been taken into account during the design phase;
– the adaptation actions are concurrent with the other events and activities of the application leading to unpredictable results;
– inadequate adaptation strategies are chosen over and over again (i.e., the adaptation enters a life-lock), wasting resources without achieving expected results.

Those kinds of adaptation-specific failures are not explicitly addressed by traditional correctness assurance techniques. Thus, novel means for the correctness assurance of adaptations are needed. In order to provide such correctness assurance techniques for adaptive SBAs, the specific aspects of one or another adaptation approach as well as the specific characteristics of the adaptation-specific failures need to be taken into account.

To pave the way towards those techniques, this paper thus sets out to

– provide a classification of existing adaptation approaches (Section 2);
– identify and classify potential adaptation-specific failures (Section 3);
– provide our vision on how to address these adaptation-specific failures (Section 4).

## 2 Adaptation in SBA

Depending on the type of adaptation, different adaptation-specific failures may occur and may have a different impact on the functionality of the application. In this section we will thus provide an overview of the different types of adaptations for SBAs. We remark that in this work we focus on the problems related to the adaptation performed automatically at run-time, since the adaptation performed at design-time may be analyzed by conventional quality assurance means, such as verification or testing.

We distinguish between configuration (or parameter) adaptation and composition adaptation. *Configuration adaptation* includes modification of certain application parameters and properties (e.g., non-functional properties) or modification of services involved in the composition (e.g., replacement of one service with another). *Composition adaptation* deals with changes in the application structure, i.e., the flow of the activities is modified. As examples, additional activities are invoked, previous operations are undone, or completely different composition fragments are executed.

Another important factor – orthogonal to the above one – concerns the way the adaptation strategy is determined and realized. In some approaches the strategy is *predefined* at design time, in other approaches the strategy is defined *dynamically* depending on the concrete situation. While the former approaches are easier to realize, they are less flexible at run-time. On the contrary, the latter approaches are more appropriate at run-time, but require complex algorithms and may be time consuming.

### 2.1 Configuration Adaptation

There exists a wide range of approaches where the changes do not affect the underlying composition model, but only certain parameters or properties of the application configuration. A typical scenario for such kind of approaches concerns the situation, where there is a change of the constituent services, i.e. the services that are bound to the application. The change may happen due to the fact that constituent services become unavailable or their quality-of-service characteristics degrade. In this case adaptation is performed by replacing "bad" services with other ones according to certain criteria. Another example of such type of changes is the modification of service-level agreements taking place as a result of re-negotiation of certain non-functional parameters.

In some of these approaches the expected configuration values or value bounds are predefined. Typically, these values are specified in service-level agreements; they define appropriate levels of quality-of-service (QoS) parameters [1,2]. The approaches aim to discover and bind the services that satisfy these constraints.

Other approaches do not specify any constraints on the parameters but try to dynamically achieve those values that provide the best possible configuration; e.g. in [3] the optimal value of several QoS metrics is targeted and in [4] the services with the best reputation are chosen.

### 2.2 Composition Adaptation

In many approaches, adaptation modifies the way, in which an application is executed, i,e., the process / composition model of the SBA is changed. These changes may be entailed by various factors, such as the necessity to customize SBA in order to deal with particular user types, or the necessity to operate in a new environment, where policies and constraints for the given SBA are different. In this case, the application should change the executed flow of activities in accordance to the adaptation specification. This may include, for example, to skip certain activities or to perform additional ones, to execute completely different process fragments, or to re-execute or roll back already performed activities.

A typical scenario for composition adaptation using predefined strategies is process migration: a new process model is defined and all the running instances should be changed in order to correspond to this model [5,6]. Adaptation in this case changes those parts of the running application instance that have not been executed yet. Other approaches define potential variants in the application model and conditions, under which the variant applies [7]. In some approaches the adaptation specification has a form of meta-level instructions to be applied to the process model, such as "redo a part of the process", "roll back to safe point", "skip to a particular execution point", "substitute a part of the process", etc [8].

There exist approaches where the composition is defined dynamically. Using certain application requirements, the composition is created automatically and may be even recomposed in case of certain changes or problems. In [9] the approach relies on a set of functional and non-functional requirements and constraints, based on which the services are composed and mediated dynamically. In [10] a domain process model with the different deviations is used as a basis for the composition.

## 3 Adaptation-Specific Failures

Adaptation-specific failures may occur – in general – for all types of adaptations, or they might be specific to particular approaches or even application domains. Moreover, they may have different character depending on the way the adaptation problem and realization are defined. This section therefore presents three major classes of adaptation-specific failures, discusses their causes, and relates them to the types of adaptation identified above.

### 3.1 Failures due to Incomplete Knowledge

When SBAs operate in a very open and dynamic environment, the list of potential events, configurations, and contexts is very broad. This makes problematic the applicability of adaptation approaches, in which the adaptation strategy is prescribed at design-time. Indeed, in this case the designer can consider only a relatively small set of situations, assuming that they are representative for all the possible variants of the execution. If this optimistic assumption is violated at run-time, the execution of the predefined adaptation specification may lead to unexpected and dangerous results. For instance, the inability to complete the adaptation (deadlocks), since the applicability of the actions is violated in a concrete context; or the possibility to end up in a "wrong" state not considered at design time.

We remark that this type of problem is specific to the approaches, where the adaptation actions or parameters are predefined.

**Configuration adaptation.** When adaptation of application configuration is considered, design-time assumptions refer to the possibility to change the application in such a way, that predefined values for the parameters are satisfied. For example, there exists a possibility to satisfy certain level of QoS properties, to discover a service with a given characteristics, etc. At run-time, however, this assumption may be violated since the values are not realistic in a given context, and adaptation may fail.

**Composition adaptation.** In case of composition adaptation, the situation is even more complex. For the process migration problem, running application instances may be in a state, where adaptation is not possible, since the resulting execution will neither correspond to the initial model nor to the final one. In case of rule-based or meta-level specifications, an implicit assumption is that the adaptation specification is correct and will successfully complete. In practice, however, the execution of adaptation activities may fail, trigger another adaptation activity, etc. Another critical situation may occur if adaptation activities are "semantically" incorrect in a given situation, i.e., they lead to a situation that is undesirable from the application logic point of view.

The situation is made more complex by the fact that adaptation specification is defined independently from the application model using specific languages and notations. This restricts applicability of traditional techniques, such as verification or testing.

### 3.2 Failures due to Concurrent Changes during Adaptation

In certain application domains possible changes or events in the application environment may be as fast as the execution of adaptation activities. In such cases, events that occur concurrently with the execution of adaptation activities may trigger another set of adaptation activities, potentially even contradictory with the initial ones.

Such interleaving of adaptation activities and contextual events executed concurrently may lead to variety of problems and inconsistencies in the application: one adaptation activity may completely "negate" the results of another one; the system may end up in an incorrect state or even deadlock; the new events continuously trigger new adaptations leading to an "adaptation stack overflow".

**Configuration adaptation.** In case of configuration adaptation, the problem refers to the changes in the corresponding configuration parameters of SBA. If the changes are so fast that they become comparable to the re-configuration time, the new "version" of the SBA may become incorrect or non-optimal. For instance, if services (or their QoS metrics) appear / disappear (or change) within minutes or seconds, newly discovered and bound service may become unavailable when invoked.

**Composition adaptation.** In case of composition adaptation, the execution flow of the application is modified and activities and tasks different from those in the original model are performed. If during the execution of these task new adaptation triggering events occur, the execution flow is modified again. Depending on how the adaptation is defined and depending on the underlying operational semantics, this may lead to a new composition implementation, where the first adaptation is not complete and therefore original requirement is not satisfied; concurrently executed processes, which may lead to incorrect state and therefore the requirements of both adaptations are violated.

Similarly to concurrent systems, the source of the problem is in the way adaptation activities are modeled, i.e., how fast, atomic, and isolated they are with respect to each other, to the system execution, and to the changes in the context.

### 3.3 Failure: Undesired Adaptation Loops

Another potential adaptation-specific failure refers to a situation, where the execution of adaptation activities directly or indirectly leads to a state, in which the same adaptation is triggered again. This situation corresponds to an adaptation *livelock*: the same adaptation activities are identified and repeated again and again without, however, any reasonable outcome.

The adaptation loop may be entailed by the problems identified above. Indeed, when the adaptation specification is defined, an implicit assumption may expect that the adaptation activities successfully complete. In practice, however, the actions may fail and the system remains in the same situation, triggering another loop of the same adaptation activities. In other cases, concurrent events may terminate the current activity and start another one (or execute it in parallel), which is the same as the initial one.

**Configuration adaptation.** Configuration adaptation loops can come in the following two forms. Firstly, re-configuration of SBA may fail and the system remains in the same state, from which the adaptation started. For example, if a service becomes unavailable, adaptation initiates discovery and binding of another service corresponding to required parameters. If these requirements can not be satisfied in the current context, adaptation fails and the activities may be started again. Secondly, if the history of adaptations (configurations) is not taken into account, it is possible that the new configuration will be equivalent to the one, for which adaptation was triggered. Accordingly, the execution of application in this new configuration may lead to the same problems, and the same adaptation will be triggered.

**Composition adaptation.** Similar problems occur, when the composition adaptation is considered. That is, adaptation activities associated to a particular situation may fail to bring the application to a new state. Consider, for instance, an example from the travelling domain: with a particular application failure (ticket is not available) one can associate the adaptation specification that requires executing an alternative path (find and book ticket using train reservation service). The execution of an alternative may end up in the same failure (no train ticket is available) and the SBA enters the undesired adaptation loop.

Similarly, the loop may take place if there are adaptation specifications that are defined independently, but have implicit mutual dependencies. Consider again the travel domain. One adaptation rule (from the user preferences) may require that if the total cost estimated for the whole trip is higher than a certain amount, nearby airports should be used instead. Another adaptation rule may enforce booking a taxi, if the airport is not within certain distance from the hotel. It is easy to see that under certain conditions, the first rule may trigger the second one, which will again require changing the airport.

## 4 Adaptation Correctness Assurance

In order to ensure that the adaptation is specified and executed correctly, the adaptation approach together with the underlying adaptation toolkit and platform should provide dedicated techniques and facilities that can uncover and mitigate the problems identified above. In this section, we first classify and discuss existing techniques for correctness assurance of service-based applications in general. Based on the capabilities of these techniques, we sketch how these can be used to address the different kinds of adaptation failures discussed above. Finally, we will highlight potential evolution of existing techniques in order to address their deficits.

### 4.1 Existing Means for Correctness Assurance

**Classification.** We will classify the techniques for correctness assurance according to the following dimensions.

First, two basic *strategies* on how to ensure correctness can be differentiated, namely constructive and analytical. *Constructive techniques* provide such form of support for the system design and execution that guarantee its correctness "by construction". That is, these techniques rely on a certain formalism and a framework, which takes the application specification and a set of predefined correctness requirements, and automatically augment it with additional facilities necessary to satisfy these requirements. *Analytical techniques* provide a possibility to analyze whether the artifacts of the system (including its specification and implementation/code) have been built in such a way as to meet certain predefined correctness properties. If potential problems are identified, the root cause for these problems is identified and the artifacts are corrected accordingly.

**Table 1.** Classification of Correctness Assurance Techniques

| *Approach* | *Strategy* | *Online / Offline* | *Configuration / Composition* |
|---|---|---|---|
| monitoring | analytical | $+/-$ | $+/+$ |
| testing | analytical | $+/+$ | $-/+$ |
| simulation | analytical | $-/+$ | $-/+$ |
| verification | analytical | $-/+$ | $-/+$ |
| model-driven development | constructive | $-/+$ | $-/+$ |
| automated configuration | constructive | $+/-$ | $+/-$ |
| automated composition | constructive | $-/+$ | $-/+$ |

Second, we can distinguish the techniques with respect to when these are employed during the life-cycle of SBAs. *Offline* techniques allow one to identify the problems before the application is put into the production mode (i.e., before the application is deployed). *Online* techniques, on the contrary, are employed while the application is executed in real settings, i.e. during the actual operation of the applications.

Third, we can distinguish between techniques that address *correctness of the configuration* (i.e., availability of services, their QoS) and techniques that address *correctness of the composition* (i.e., application behavior is correct).

**Techniques.** Table 1 provides an overview of the existing major approaches for correctness assurance in SBAs together with the capabilities they provide:

– *Monitoring* observes the service-based application during its *current* execution (i.e., actual operation) in order to detect deviations from the expected behavior. This includes monitoring QoS properties [11], assertions [12], or complex behavioral properties of compositions [13,14].
– The goal of *testing* is to (systematically) *execute* services or service-based applications with predefined inputs and to observe the outputs in order to uncover failures; Examples for testing techniques include [15,16].
– *Simulation* corresponds to testing of a composition specification without actually executing services [17];
– During *verification*, artifacts (e.g., a service specification) are systematically *examined* (without execution) in order to ascertain that some predefined properties are met. There exist approaches that use model checking (e.g., [18,19]) or logic-based techniques [20,21].
– *Model-driven development* aims at generating low-level specifications (closer to the implementation) given high-level models and additional transformation rules that preserve certain properties [22,23].
– During *automated configuration* a predefined abstract composition model is automatically filled with the concrete services or parameters in order to satisfy some criteria (e.g., optimization of QoS metrics) [24];
– During *automated service composition*, services are composed according to the goal of the user / designer (e.g., provide a composed travel organizer from flight, hotel, train booking services) together with additional constraints, like transactionality constraints (i.e., do not book hotel if the flight booking fails); examples for automated service composition frameworks include [25,26].

## 4.2 Dealing with Adaptation Failures

The techniques discussed above may also be applied to tackle the failures that are specific to the adaptation. These failures, however, pose new specific requirements, which may not be addressed with the existing techniques, and would need their evolution or even new approaches. For each of the adaptation-specific failure described in Section 3 we will discuss their requirements and the applicability of the existing approaches in these regards.

**Failures due to incomplete knowledge.** In the case of configuration adaptation, the problem refers to the violation of the design-time assumptions about the application configuration properties (e.g., inability to find services satisfying predefined QoS levels). In case of composition adaptation, the execution of a predefined adaptation strategy leads to unexpected and incorrect behavior, since not all the possible settings and contexts are considered at design time.

To address these problems, one of the following requirements should be met:

- (1) avoid using predefined ("hard-coded") specifications but adapt them to concrete run-time situations or contexts;
- (2) validate the realizability of the specifications before the execution / deployment of an application;

For what concerns configuration adaptation, requirement (1) may be achieved with existing techniques, such as automated application configuration through dynamic negotiation of SLAs, while requirement (2) needs novel approaches as it follows from Table 1.

For what concerns composition adaptation, the first requirement may be achieved by extending existing automated composition or model-driven techniques, in order to accommodate for potential run-time changes and events. Analogously, existing verification and simulation techniques require further extensions in order to validate adaptation specifications. In particular, there is a need to formalize the semantics of the adaptation languages and relate it to the semantics of the underlying application models; it is necessary to define "correctness" criteria for the adaptation execution; it is necessary to model and adequately represent the dynamics of the execution context.

**Failures due to concurrent changes during adaptation.** When the changes and relevant events occur concurrently with the execution of adaptation activities, the adaptation may become redundant, have undesirable effects or even failures. In order to prevent these problems, the adaptation framework should take potential changes and events into account when the adaptation strategy is determined. In particular, the following requirements should be addressed:

- (1) analyze the dynamics of the execution context of the application (i.e., frequency and timing of potential changes, duration of application activities);
- (2) analyze the impact of these factors on the adaptation execution in order to guide the development of proper adaptation specifications.

Existing correctness techniques (i.e., monitoring or verification) do not provide appropriate means to address those requirements and therefore should be extended. Indeed, it

**Table 2.** Classification of Advanced Correctness Techniques

| *Approach* | *Strategy* | *Online/Offline* | *Config/Compos* | *Incompl. knowl.* | *Concur. change* | *Adaptation loop* |
|---|---|---|---|---|---|---|
| Offline adaptation analysis | analytical | $-/+$ | $-/+$ | $+$ | $+$ | $+$ |
| Pre-deployment monitoring and testing | analytical | $-/+$ | $+/+$ | $+$ | $+$ | $-$ |
| Online verification and simulation | analytical | $+/-$ | $-/+$ | $+$ | $+$ | $+$ |
| Online automated composition | constructive | $+/-$ | $-/+$ | $+$ | $-$ | $-$ |
| Built-in adaptation | constructive | $-/+$ | $-/+$ | $+$ | $+$ | $+$ |
| Monitoring adaptation history | analytical | $+/-$ | $+/+$ | $-$ | $-$ | $+$ |
| Stability metrics | constructive | $+/-$ | $+/-$ | $-$ | $+$ | $-$ |

is necessary not only to observe the relevant events and changes, but also to monitor how frequent these changes are with respect to the application activities. This information should then be modelled and used in the analysis, and, consequently, when adaptation specifications are derived.

**Failure: Undesired adaptation loops.** When for some reason adaptation fails or brings the application to a situation from which it has been initiated, the same set of adaptation activities may be initiated. As a result, the application enters the “adaptation loop”, thus consuming resources without any effect. In order to avoid this problem, the information about previous situations and triggered adaptations should be considered when the adaptation strategy is determined. That is,

- (1) define adaptation in such a way that the loop can not appear, or
- (2) define a special mechanisms to leave the loop when it is detected.

Neither in case of composition adaptation nor in case of configuration adaptation, existing correctness techniques are enough to address these requirements. Indeed, analytical (respectively, constructive) techniques require specific means to check the specification against such loops (resp., to construct loop-free specifications and executions).

### 4.3 Advanced Correctness Techniques

In order to tackle the failures that are specific to the adaptation of SBA, the existing correctness assessment techniques are not applicable in the way they are used for static applications. These techniques require specific extensions, and, moreover, in certain cases novel approaches are necessary. Here we highlight future potential techniques that could be applied in combination with the SBA adaptation approaches. Table 2 classifies advanced techniques and maps them to the adaptation-specific failures.

**Offline adaptation verification and simulation.** The approach to extend conventional techniques with the ability to verify application specification in combination with the adaptation specification. This requires $(i)$ representing the latter using the same formalism as for the application; $(ii)$ representing the dynamics of the environment (changes and events) that are relevant for the adaptation; and $(iii)$ defining specific properties to be checked by the analysis tool. In case of problem due to incomplete knowledge, this may include the necessity to complete the adaptation execution or an ability to reach some "stable" or "correct" state. In case of adaptation loop problem, the property may express absence of such loops. In case of concurrent changes, the model and properties may express timed constraints on the adaptation behavior [27].

The most challenging problem here is to model and represent the behavior of the environment, which is often difficult to foresee for open systems, or may generate behaviors that never happen in practice.

An initial step towards offline adaptation analysis has been presented by Fugini et al. [28]. They present an approach for testing SBAs extended with fault injection mechanisms. The goal of the approach is to check if and how the application reacts to certain faults, such as delays or data errors.

**Pre-deployment monitoring and testing.** Here the idea is to evaluate some properties and metrics of the application and its environment before the system is ready for deployment. This may include, in particular, monitoring the QoS of potential services, evaluating characteristics of the context, estimating duration of application activities, etc. Furthermore, this kind of analysis may be applied in order to evaluate the dynamics of the application context, e.g., how often relevant changes happen, what are the ranges, etc. This information may be used to restrict the models of the environment exploited by the previous approach and, therefore, to make this models more compact and realistic.

**Online verification and simulation.** This technique consist of verifying or simulating the adaptation specification at run-time before its actual execution. Similarly to the offline adaptation analysis, this technique permits verifying correctness of the specification. On the positive side, it is applied in a concrete situation, and therefore is simpler to model and represent. On the downside, such analysis may slow down the application adaptation.

A sort of first attempt towards online verification of adaptation has been presented in [29]. It aims to verify correctness constraints in the scope of process migration problem. When the running process instance is dynamically changed, the proposed technique is used to check that the changes are compatible with certain constraints.

**Online automated composition.** This approach aims to bring the automated techniques applied at design-time to the execution phase. In this way, the adaptation specification is defined dynamically, taking into account concrete situation. However, as in the case of online analysis, such technique may considerably slow down the adaptation.

**Built-in adaptation.** This approach combines model-driven techniques with the automated composition approach. The adaptation specification defined at design-time is automatically composed with (i.e., built-in) the application specification. As a result, an integrated executable specification is generated that takes into account possible run-time changes and events. As in the case of offline adaptation analysis, the need to model possible events increases the complexity of the technique. On the positive side, the resulting specification is efficient with respect to online composition.

**Monitoring adaptation history.** One way to deal with the adaptation loop problem is to monitor and store the information about previous adaptations (adaptation history).

In particular, this information includes the event/situation, in which the adaptation was triggered, and the outcome of its execution (i.e., positive/negative, state reached, etc.). When the need for adaptation is identified, the system will compare the situation with previous histories. If the situation was already registered and the previous adaptation failed, then in the current situation some other strategy should be applied.

**Stability metrics.** In order to deal with highly dynamic environments, in which the changes may happen during adaptation, one can use special stability metrics. This metrics may be used in order to estimate and keep information on how stable a certain property is over time, for example, the frequency of changes in certain QoS parameter of the service involved in the composition. Such metrics would allow one to discover and bind only "stable" services, such that the adaptation is not triggered too often. Indeed, this requires specific monitoring techniques, as well as the corresponding capability of, e.g., the discovery framework.

## 5 Conclusions

Adaptive service-based application are often subject to specific failures and problems that are not exposed in "static" systems. In order to address these problems, novel approaches that extend both adaptation and traditional correctness assessment means are necessary. In this paper we have identified and classified adaptation-specific failures; we have also demonstrated how these failures show themselves in different adaptation approaches. Through a review and a classification of the existing correctness assurance techniques we have demonstrated that these techniques are not enough to deal with adaptation specific failures. Based on the identified gaps and requirements, we have revealed future directions and approaches that would improve correctness in the adaptive service-based applications. As a future work we would like to instantiate the proposed approaches and to integrate them within concrete adaptation frameworks.

## References

1. Al-Ali, R.J., Hafid, A., Rana, O.F., Walker, D.W.: QoS Adaptation in Service-Oriented Grids. In: Middleware Workshops, pp. 200–210 (2003)
2. Canfora, G., Penta, M.D., Esposito, R., Villani, M.L.: An Approach for QoS-aware Service Composition Based on Genetic Algorithms. In: Proceedings of the 2005 conference on Genetic and evolutionary computation, pp. 1069–1075 (2005)
3. Chafle, G., Dasgupta, K., Kumar, A., Mittal, S., Srivastava, B.: Adaptation in Web Service Composition and Execution. In: Int. Conference on Web Services - ICWS, pp. 549–557 (2006)
4. Bianculli, D., Jurca, R., Binder, W., Ghezzi, C., Faltings, B.: Automated Dynamic Maintenance of Composite Services based on Service Reputation. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749, pp. 449–455. Springer, Heidelberg (2007)
5. Reichert, M., Rinderle, S., Kreher, U., Dadam, P.: Adaptive process management with adept2. In: ICDE, pp. 1113–1114 (2005)

6. Hallerbach, A., Bauer, T., Reichert, M.: Managing Process Variants in the Process Life Cycle. Technical report, University of Twente, TR-CTIT-07-87 (2007)
7. Siljee, J., Bosloper, I., Nijhuis, J., Hammer, D.: DySOA: Making Service Systems Self-adaptive. In: Benatallah, B., Casati, F., Traverso, P. (eds.) ICSOC 2005. LNCS, vol. 3826, pp. 255–268. Springer, Heidelberg (2005)
8. Baresi, L., Guinea, S., Pasquale, L.: Self-healing BPEL processes with Dynamo and the JBoss rule engine. In: ESSPE 2007: Int. workshop on Engineering of software services for pervasive environments, pp. 11–20 (2007)
9. Verma, K., Gomadam, K., Sheth, A.P., Miller, J.A., Wu, Z.: The METEOR-S Approach for Configuring and Executing Dynamic Web Processes. Technical report (2005)
10. Lazovik, A., Aiello, M., Papazoglou, M.P.: Associating Assertions with Business Processes and Monitoring their Execution. In: Service-Oriented Computing - ICSOC 2004, Second Int. Conference, pp. 94–104 (2004)
11. 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)
12. 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)
13. Mahbub, K., Spanoudakis, G.: Monitoring WS-Agreements: An Event Calculus-Based Approach. In: Baresi, L., Nitto, E.D. (eds.) Test and Analysis of Web Services, pp. 265–306. Springer, Heidelberg (2007)
14. Barbon, F., Traverso, P., Pistore, M., Trainotti, M.: Run-Time Monitoring of Instances and Classes of Web Service Compositions. In: Int. Conference on Web Services (ICWS), pp. 63–71 (2006)
15. Bai, X., Chen, Y., Shao, Z.: Adaptive Web Services Testing. In: 31st Annual Int. Computer Software and Applications Conference (COMPSAC), vol. 2, pp. 233–236 (2007)
16. Canfora, G., di Penta, M.: SOA: Testing and Self-checking. In: Proceedings of Int. Workshop on Web Services - Modeling and Testing - WS-MaTE, pp. 3–12 (2006)
17. Mayer, P., Luebke, D.: Towards a BPEL Unit Testing Framework. In: Workshop on Testing, Analysis, and Verification of Web Services and Applications, TAV WEB 2006, pp. 33–42 (2006)
18. Fu, X., Bultan, T., Su, J.: Analysis of Interacting BPEL Web Services. In: Proceedings of the 13th Int. World Wide Web Conference (WWW 2004) (2004)
19. Sharygina, N., Krning, D.: Model Checking with Abstraction for Web Services. In: Test and Analysis of Web Services, pp. 121–145 (2007)
20. Davulcu, H., Kifer, M., Ramakrishnan, I.V.: CTR-S: A Logic for Specifying Contracts in Semantic Web Services. In: Proc. WWW, pp. 144–153 (2004)
21. Grüninger, M.: Applications of PSL to Semantic Web Services. In: Proc. 1st Int. Workshop on Semantic Web and Databases, pp. 217–230 (2003)
22. Skogan, D., Gronmo, R., Solheim, I.: Web Service Composition in UML. In: Proceedings of the Enterprise Distributed Object Computing Conference (EDOC), pp. 47–57 (2004)
23. Castro, V.D., Marcos, E., Sanz, M.L.: A Model-Driven Method for Service Composition Modelling: a Case Study. Int. J. Web Eng. Technol. 2, 335–353 (2006)
24. Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., Sheng, Q.Z.: Quality Driven Web Services Composition. In: WWW 2003 (2003)
25. Marconi, A., Pistore, M., Poccianti, P., Traverso, P.: Automated Web Service Composition at Work: the Amazon/MPS Case Study. In: Int. Conference on Web Services (ICWS), pp. 767–774 (2007)

26. Berardi, D., Calvanese, D., De Giacomo, G., Mecella, M.: Composition of Services with Nondeterministic Observable Behavior. In: Benatallah, B., Casati, F., Traverso, P. (eds.) ICSOC 2005. LNCS, vol. 3826. Springer, Heidelberg (2005)
27. Kazhamiakin, R., Pandya, P.K., Pistore, M.: Representation, Verification, and Computation of Timed Properties in Web Service Compositions. In: Proceeding of the Int. Conference on Web Services (ICWS), pp. 497–504 (2006)
28. Fugini, M.G., Pernici, B., Ramoni, F.: Quality Analysis of Composed Services through Fault Injection. In: Business Process Management Workshops, pp. 245–256 (2007)
29. Ly, L.T., Rinderle, S., Dadam, P.: Integration and Verification of Semantic Constraints in Adaptive Process Management Systems. Data Knowl. Eng. 64, 3–23 (2008)

# A Service Based Development Environment on Web 2.0 Platforms

Xabier Larrucea[1], Rafael Fernandez[2], Javier Soriano[2], Andrés Leonardo Martínez[3], and Jesus M. Gonzalez-Barahona[4]

[1] European Software Institute, Parque Tecnologico de Zamudio 204, 48170 Zamudio, Spain
Xabier.Larrucea@esi.es
[2] Computer Networks & Web Technologies Lab., School of Computing, Universidad Politecnica de Madrid, 28660, Boadilla del Monte, Madrid, Spain
{rfernandez,jsoriano}@fi.upm.es
[3] Telefonica Research & Development, 28043 - Emilio Vargas 6, Madrid, Spain
almo@tid.es
[4] GSyC/LibreSoft, Universidad Rey Juan Carlos
jgb@gsyc.escet.urjc.es

**Abstract.** Governments are investing on the IT adoption and promoting the so-called e-economies as a way to improve competitive advantages. One of the main government's actions is to provide internet access to the most part of the population, people and organisations. Internet provides the required support for connecting organizations, people and geographically distributed developments teams. Software developments are tightly related to the availability of tools and platforms needed for products developments. Internet is becoming the most widely used platform. Software forges such as SourceForge provide an integrated tools environment gathering a set of tools that are suited for each development with a low cost. In this paper we propose an innovating approach based on Web2.0, services and a method engineering approach for software developments. This approach represents one of the possible usages of the internet of the future.

**Keywords:** Service, Web2.0, Method engineering.

## 1 Enterprise 2.0 Technologies and Quality Assurance

Nowadays organisations are worried about mainly two main issues: collaboration and quality assurances. As global market opportunities and competition increase, collaboration is becoming more and more essential for improving productivity and accelerating innovation at the personal, team, group, enterprise and business coalition levels. Many enterprise collaboration platforms have already been developed and successfully deployed in both large, and small- and medium-sized enterprises (SMEs). Enterprise collaboration has recently come to benefit from the emergence of an enterprise-oriented specialization of the Web 2.0 vision, commonly referred to as

© Springer-Verlag Berlin Heidelberg 2008

Enterprise 2.0 [1] providing new models and tools for emergent collaboration and co-creation. Enterprise collaboration is thus being enhanced by virtual communities that leverage social linking and tagging tools (like tools for social networking, social bookmarking and social search), user-contributed content management platforms (like enterprise Wikis, blogs and forums), tools that leverage user opinions (like tools supporting comments and voting), subscription-based information distribution tools (like Enterprise RSS feeds), etc. Used in the context of a carefully engineered collaboration strategy, these technologies provide a wealth of collaborative services for software developers [2].

On the other side quality is still representing a nightmare for too many organizations. In fact quality cost is one of the most important considerations in software production [3], [4]. Quality assurance practices and software products quality represent in most of cases the forgotten requirement and it is becoming a hard task to select an appropriate infrastructure in order to fulfil at the same time customers' requirements and some level of quality assurance. The resulting solutions are usually defined in terms of what functionalities are exposed and the question about "what is the quality required?" and "How do we achieve this quality?" are effaced from stakeholders' memory.

This happening generalises a broad range of situations such as consultancy, in-house and outsourcing developments. The evolution of Internet technologies such as Web2.0 and mash up platforms are supporting collaboration mechanisms but at the same time they need to fulfil quality models requirements (e.g., Capability Maturity Model Integrated-CMMI) [5]. In order to facilitate the fulfilment of these two challenges the proposed architecture combines the evolution of these new Enterprise 2.0 technologies and quality assurance facilities. In fact the resulting collaborative environment is enriched with the savoir-faire (knowledge management) in software production environments.

## 2 Ezforge: New Generation of Networked Forges Supporting Collaborative Software Development

Organizations tend to behave like dynamically reconfigurable networked structures that carry out their tasks by means of collaboration and teamwork. Effective teamwork is an essential part of any non-trivial engineering process, and collaborative capabilities are an essential support for these teams. Software development is no exception; it is in itself a collaborative team effort, which has its own peculiarities. Both in the context of open source software development projects and in organizations that develop corporate products, more and more developers need to communicate and liaise with colleagues in geographically distant areas about the software product that they are conceiving, designing, building, testing, debugging, deploying and maintaining. In their work, these development teams face significant collaborative challenges motivated by barriers erected by geographic distances, time factors, number of participants, business units or differences in organizational hierarchy or culture that inhibit and constrain the natural flow of communication and collaboration. To successfully overcome these barriers, these teams need tools by means of which to communicate with each other and coordinate their work. These tools should also take into account the functional, organizational, temporal and spatial characteristics of this collaboration. Software product users are now becoming increasingly involved in this process, for which reason they should also be considered. In response to this necessity, forges are gaining importance both in the open source context and the corporate environment.

Following the ideas in [6], a forge can be described as a kind of collaborative development environment (CDE) that provides a virtual space wherein all the stakeholders of a software development project –even if distributed by time or distance - may negotiate, brainstorm, discuss, share knowledge, and labor together to carry out a software product and its supporting artifacts. It integrates multiple collaborative tools and resources, thanks to which it offers a set of services to aid all the stakeholders in the software development area, including managers, developers, users, commercial software manufacturers and software product support enterprises, to communicate, cooperate and liaise. Forges consider software development's social nature and assure that the people who design, produce, maintain, commercialize and use software are aware of and communicate about the activities of the others simply, efficiently and effectively, also encouraging creativity and driving innovation. In doing so, forges provides with a safe a centralized solution conceived to optimize collaborative and distributed software development generally based on Internet Standards. This solution serves a number of essential purposes, including:

- A holistic integration of disparate collaborative processes and tools through a collaborative environment,
- an expansion of visibility and change control,
- a centralization and administration of resources, and
- a reinforcement of collaboration, creativity and innovation.

## 3 EzForge

The appearance of Enterprise 2.0-based forges, such as EzForge [7], enable software development teams to find, customize, combine, catalogue, share and finally use tools that exactly meet their individual demands. Supported by the EzForge platform, they can select and combine development tools hosted by third parties rather than buying a pre-determined, inflexible and potentially heavyweight software development environment.

EzForge as the main part of the proposed architecture is based on the idea of considering forges not as single sites providing a monolithic set of services to host projects as isolated silos of knowledge, but as a collection of distributed components providing services among which knowledge is shared. Each project decides on its own customized set of services, and users can configure their own mashup-based user interface. Fig. 1 depicts the 3-tier EzForge architecture.

Back-end tier is where integrated and legacy systems reside. It is important to take into account that the EzForge architecture imposes no limitations to where the different components may be hosted. Those systems have their own set of basic forge services, such as source code management, wiki, and issue/bug tracking services, and they are integrated into the forge following a Web 2.0 approach consisting of

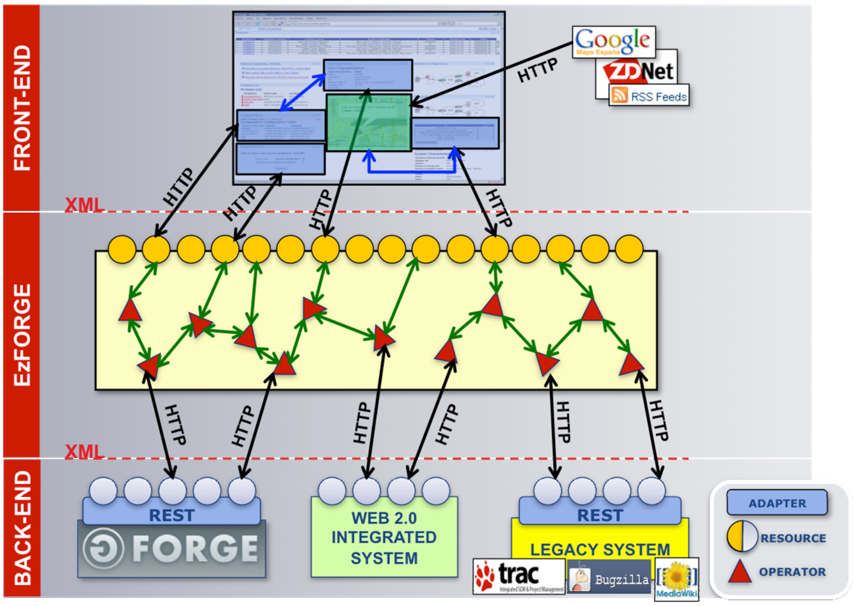


**Fig. 1.** EzForge architecture

transforming their legacy services into a uniform layer of resources. These resources are components designed following the REST architectural style [8] that can be accessed through an URI via HTTP. Integrated systems already follow this approach, while legacy systems needs adapters to perform this task. Thanks to the aforementioned layer of resources, the EzForge tier can access them to gather and process their data by means of special resources called operators, elements designed to get data from resources and use it to produce new data that can be processed by other resources, enabling their remix and composition. Doing so, EzForge creates the set of resources the forge will deliver to its final users.

Once the EzForge tier has its forge resources set, final users are empowered by allowing them to design their own front-end layer (or forge user interface) by means of composing user interface building blocks called gadgets, which are endowing with the forge resources. Following this approach, users can mix and compose forge resources on their own, allowing them to choose the best resources to meet their needs. User can even include external resources, such as Google Maps or RSS feeds, into their UI, using all of them as a whole. They will use whichever resources they like to create ad hoc instant forge UI, encouraging resources mashup, and following the DIY (“do it yourself”) philosophy.

## 4 Savoir-Faire in Software Production Environments

More often than we can imagine, developers are plunged in the ocean of tools and procedures required in their daily work. Until now we have defined a forge (EzForge) as a development platform but project responsibilities are delegated to developers and the management of developers' know-how is not taken into account. How do we materialize the know-how of your developments? How can we assure that our software products are developed as defined by the organization? These questions represent some of the factors that guide organizations to consider the materialization of their know-how and of their internal procedures in somehow for helping organisations to avoid or overcome barriers and hurdles raised during their work. For example some of these elements are the integration of new developers within development teams and quality assurance with respect to the requirements of quality models such as CMMI®.

This is a cornerstone in our software developments and it is part of the knowledge management (KM) broached by our architecture. One of the competitive advantages for organizations is their know-how, their human capital. Therefore we need to make explicit tacit knowledge in order to share information and to promote the savoir-faire within the organizations. In this sense in the area of KM Peter M. Senge [9] defines the learning organizations and he states five interrelated disciplines for the creation of smart and competitive organizations. In our approach we have used method engineering approach as the way to make explicit tacit software production processes and methods in order to spread knowledge within the organization.

Method engineering approach [10] is used for several software developments and approaches [11], [12] and we have applied this approach in our context. In fact we have adopted Software Process Engineering Metamodel (SPEM) 2.0 [12] as a language for the definition of software development processes and by the Eclipse Process Framework (EPF) (www.eclipse.org/epf),as a tool support for defining processes and methods in a Eclipse-based environment. The main idea is to define a methodology and relates method elements EzForge resources required for the software development. The huge number of resource-oriented services that are envisioned to be available in an Internet-scale networked forge will become unmanageable and thus useless for its users. Even if a repository service is provided, it will eventually become difficult for software development stakeholders to find out which resources (i.e. tool services) are appropriate for their development process.

This is the reason why we have created dedicated catalogues. In fact they provides navigation services for software development stakeholders and help them to find out which resources (i.e. tool services) they need to create the mash-ups they want. EzForge provides a user-contributed, "living" catalogue of resources founded on the Web 2.0 vision for user co-production and harnessing of collective intelligence (see Fig. 2. ). This would provide all stakeholders with a collaborative semantic Wiki, and tagging and searching-by-recommendation capabilities for editing, remixing and locating resources of their interest.

The catalogue sets out the knowledge available within a certain community for composing resources (e.g. a method from its fragments) in a graphical and intuitive fashion and for sharing them in a world-wide marketplace of forge services.

The catalogue allows users to create complex mash-up solutions by just looking for (or being recommended) "pre-cooked" or off-the-shelf resources and customizing these resources to suit their personal needs and/or the project requirements, interconnecting resources, and integrating the outcome in their development workspace. These decisions are defined during the development process definition.

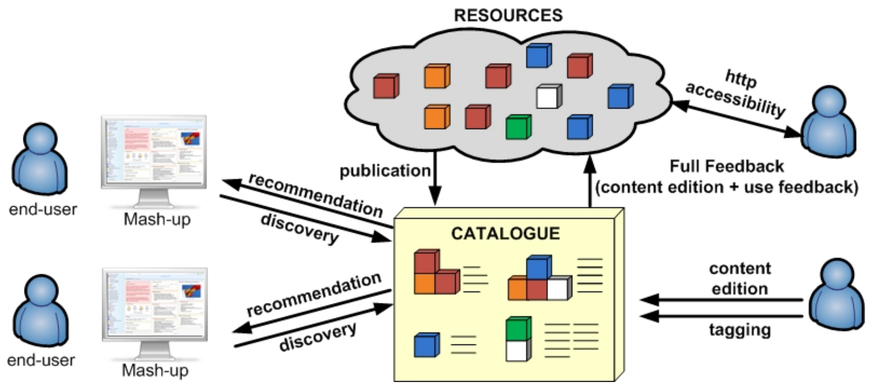


**Fig. 2.** Cataloguing Resources

"Folksonomies" of user-created tags will emerge, grow as users add information over time and act as important facilitators of a useful marketplace of resources for the networked forge.

Earlier approaches to service discovery and description like UDDI are not adequate to support human beings in easy resource retrieval and evaluation. By contrast, the exploitation of collective intelligence and user-driven resource categorization is beneficial for users.

A straightforward application of our savoir-faire approach using the catalogue is split in four steps:

- Evaluation of new developments: taking into account previous experiences, method engineers evaluate a new software development. In this phase, Knowledge Management plays a relevant role identifying software development phases, tasks and problems that are resident in developers' minds. Method engineers should evaluate previous experiences and clearly specify what objectives of this new development are.
- Selection of method fragments based on previous experiences: method engineers select the appropriate set of method fragments fitting software requirements. In fact in this context each method fragment is related to a set of Web2.0 resources. A basic catalogue contains the relationships between software processes and Web2.0 resources. Each task is related to workproducts representing a resource and therefore method engineers could specify the appropriate tools support at each software development stage.
- Composition of method fragments in order to produce a software development process used in the organization. In this step the selected method fragments are composed defining a flow that it is guided by the methodology. This composition states which are the selected resources at each stage of the development process. This approach is similar to Business Process Execution Language (BPEL) where Web Services are called following a specific order and sequence.

- Deployment within the organization. The resulting software development process is represented as a model and it is used by our forge. At this step and following CMMI® terminology, the result represents a defined and managed Standard Software Process (SSP) for an organization. This is a requirement for organizations aiming to achieve compliancy with CMMI® level 3.

This novel approach uses a method engineering approach in order to make explicit the savoir-faire in software developments within an organization (see Fig. 3.).A catalogue contains relationships between method fragments represented by the methodology using SPEM2.0, and resources that are represented within the forge as aggregators or connectors. Method engineers select the required method fragments needed for their software developments. In this context they select indirectly a set of aggregators and/or connectors that are related to specific resources. These resources are the basic tools within the development environment. Therefore we are reducing the gap between methodologies and software development tools support. This process allows the customisation of the resources and therefore the user's interfaces.

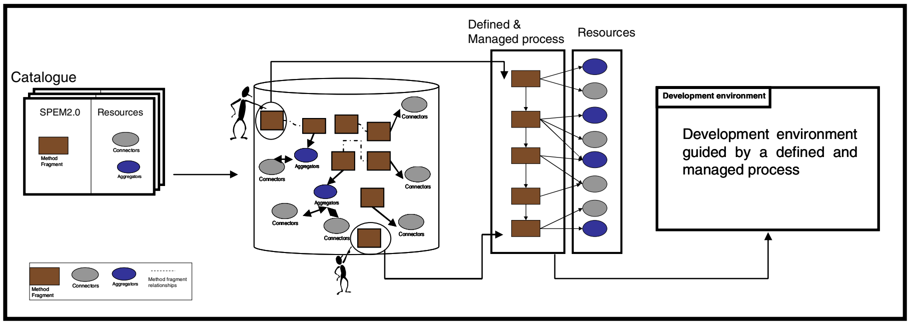

Catalogue
SPEM2.0
Resources
Method Fragment
Connectors
Aggregators
Defined & Managed process
Resources
Development environment
Development environment guided by a defined and managed process
Method fragment relationships

**Fig. 3.** A developer's environment customisation

## 5 Method Engineering and EzForge Architecture: A Holistic View

EzForge is a highly configurable and extensible user-centric collaborative software development tool that follows a novel mashup-based lightweight approach, given by the EzWeb core technology. Its user interface is defined by the user himself, who is able to make it up by assembling a set of small web applications called gadgets, which are the face of the services being offered by the forge. Up to now, there have been several attempts to bring mashup-based tools to the organizations [14], with satisfactory results.

But with regard to software development, open source development tools don't take into account a key point in the software development within organizations: quality.

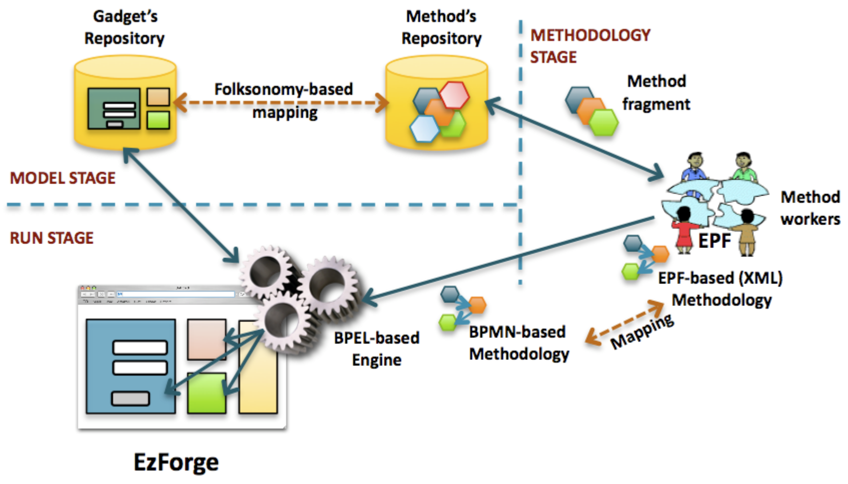


**Fig. 4.** A holistic view of Method Engineering and EzForge

Method Engineering and EzForge architecture (Fig. 4.) is compatible with the savoir-faire process defined previously and technically it is defined into three levels:

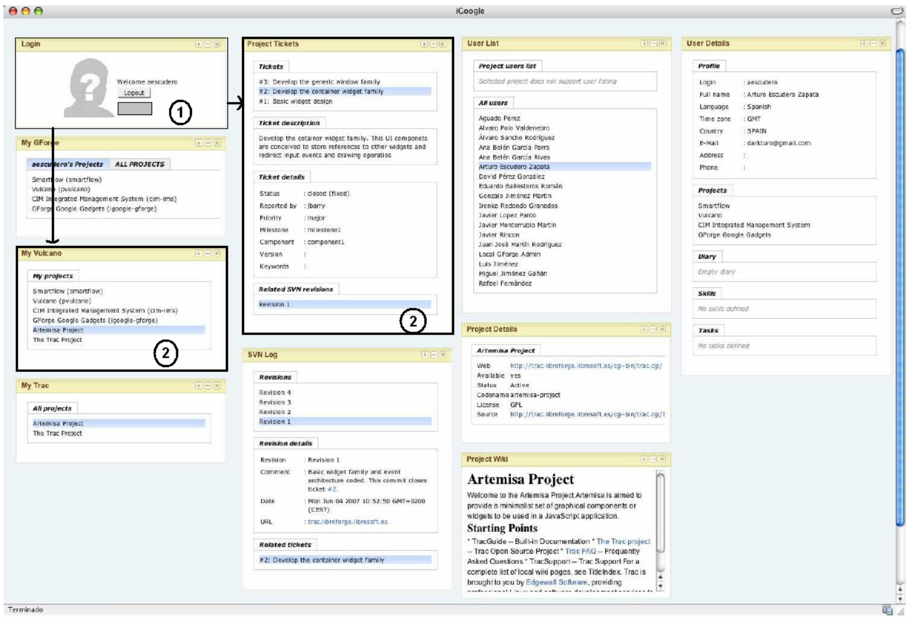
**Model stage.** The goal of this stage is to link the available gadgets from the EzForge catalogue with the method fragments that exist in the method repository and that will conform lately the used methodology. This catalogue provides access in an automated way to EzForge catalogue in the execution level. For this purpose, we have developed a folksonomy-based mapping, which allows us to create that link by using social tagging techniques. By using these tags we will be able to choose the gadget or gadget group labelled with the method identifier in methodology run time in an easy way. Besides, it gives us a way to incorporate the organization internal knowledge about how things work better, as it is their own developers who carry out this tagging process.

**Methodology stage.** It is in between model and run stages, and as we said before, it is where method workers select the method fragments that will make up the organization's methodology. To do so, method workers use the Eclipse Process Framework, which helps us to get, among other things, an XML representation of the methodology.

**Run stage.** Once we've got the methodology, the next step is to put it in execution by means of a workflow engine. Thanks to this, EzForge can choose the appropriate and required development tools depending on the ongoing development phase.

Thus, our proposed infrastructure takes the advantages of method engineering and brings them all to EzForge, allowing companies and organizations to have a user-centric collaborative development tool which can guide its users through the development process.

At this level the resulting and running application, Fig. 5. is shown using a web browser. Gadgets presented in this interface are those that have been defined by the method engineer when he was defining the organization's standard software process.



Welcome aescudero

aescudero's Projects | ALL PROJECTS
Smartflow (smartflow)
Vulcano (pvulcano)
CIM Integrated Management System (cim-ims)
GForge Google Gadgets (igoogle-gforge)

My projects
Smartflow (smartflow)
Vulcano (pvulcano)
CIM Integrated Management System (cim-ims)
GForge Google Gadgets (igoogle-gforge)
Artemisa Project
The Trac Project

All projects
Artemisa Project
The Trac Project

Tickets
#3: Develop the generic window family
#2: Develop the container widget family
#1: Basic widget design

Ticket description
Develop the cotainer widget family. This UI componets are conceived to store references to other widgets and redirect input events and drawing operatios

Ticket details

| | |
|---|---|
| Status | : closed (fixed) |
| Reported by | : jbarry |
| Priority | : major |
| Milestone | : milestone1 |
| Component | : component1 |
| Version | : |
| Keywords | : |

Related SVN revisions
Revision 1

Revisions
Revision 4
Revision 3
Revision 2
Revision 1

Revision details

| | |
|---|---|
| Revision | : Revision 1 |
| Comment | : Basic widget family and event architecture coded. This commit closes ticket #2. |
| Date | : Mon Jun 04 2007 10:52:50 GMT+0200 (CEST) |
| URL | : trac.libreforge.libresoft.es |

Related tickets
#2: Develop the container widget family

Project users list
Selected project does not support user listing

All users
Aguado Pérez
Alvaro Polo Valdenebro
Álvaro Sancho Rodriguez
Ana Belén García Parra
Ana Belén García Rivas
Arturo Escudero Zapata
David Pérez González
Eduardo Ballesteros Román
Gonzalo Jiménez Martín
Irenka Redondo Granados
Javier Lopez Pardo
Javier Monterrubio Martin
Javier Rincon
Juan José Martín Rodríguez
Local GForge Admin
Luis Jiménez
Miguel Jiménez Gañán
Rafael Fernández

Artemisa Project

| | |
|---|---|
| Web | http://trac.libreforge.libresoft.es/cgi-bin/trac.cgi/ |
| Available | yes |
| Status | Active |
| Codename | artemisa-project |
| License | GPL |
| Source | http://trac.libreforge.libresoft.es/cgi-bin/trac.cgi/t |

Artemisa Project
Welcome to the Artemisa Project Artemisa is aimed to provide a minimalist set of graphical components or widgets to be used in a JavaScript application.
Starting Points
* TracGuide -- Built-in Documentation * The Trac project -- Trac Open Source Project * Trac FAQ -- Frequently Asked Questions * TracSupport -- Trac Support For a complete list of local wiki pages, see TitleIndex. Trac is brought to you by Edgewall Software, providing

Profile

| | |
|---|---|
| Login | : aescudero |
| Full name | : Arturo Escudero Zapata |
| Language | : Spanish |
| Time zone | : GMT |
| Country | : SPAIN |
| E-Mail | : darkturo@gmail.com |
| Address | : |
| Phone | : |

Projects
Smartflow
Vulcano
CIM Integrated Management System
GForge Google Gadgets

Diary
Empty diary

Skills
No skills defined

Tasks
No tasks defined

**Fig. 5.** Runtime execution overview

Obviously there are some permanent gadgets in this interface but most of the gadgets are configured during the model and methodology levels. Once we start/continue a software development, these gadgets are modified accordingly to a software development phase. In Fig. 5. the main gadget marked as "1" acts upon the existence of gadgets marked as "2". Moreover there are other relationships amongst gadgets as they are highlighted in this figure. When we select an element in one gadget automatically there is a selection of the related elements in other gadgets. These relationships are not specified by the defined and managed methodology, they are implemented by the forge.

## Why this approach covers quality practices?

Managers are not worried about the selected architecture but they are more focused on costs and quality requirements used for the developments carried out in their organizations. Method engineering & EzForge architecture combines quality practices and a development infrastructure based on Web2.0 assuring quality aspects, with low cost, open to new OS tools, it represents an integrated tool and it overcomes new developer's barriers.

## But why this approach covers quality practices?

CMMI® is one of most used quality reference model and it comprises two representations: staged and continuous. Whatever CMMI® representation stakeholders select for their adoption, there is a common problem: a separation between process areas due to a scarce tool support from a holistic perspective. Nowadays engineering practices and process/project practices are separated and one of the main tasks for adopters is to assure that all process areas are coherent and consistent among them. Method engineering & EzForge architecture assures quality practices because the process defined and managed is used accordingly to its specification, and the development forge is guided by the methodology designed. In addition it also covers engineering and support process area because it provides an integrated development environment gathering requirements and configuration managements.

## 6 Conclusions

The presented approach combines method engineering and Web2.0 technologies in order to create a new generation of software developments tools and methods. Our approach starts from an explicit definition of the main tasks that a developer should carry out and its development environment is modified with respect to the organisation's development process. This novel approach combines an extendable development environment based on Web2.0 technologies with quality practices and tools. As results, we reduce the gap between development tools with low cost; we implement an extendable environment where we can select the appropriate tools support, and quality practices and tools are assured by this service based architecture. Web2.0 technologies relevance is becoming during these last years a new wave in several environments and method engineering approach provides architecture to make explicit the knowledge managed by organizations. Our experience combining both approaches places us on the road for a new tool generation of software developments which benefits are:

- Facilitate collaboration in heterogeneous contexts: Web2.0 technologies facilitate software developments on the Web.
- User interface configuration: another advantage that comes directly from using method engineering and EzForge altogether is the possibility of generating the user interface based either on the methodology used. Thus, when creating a new project on the forge, it will be able to help the user to choose the tools to be used
- Help developers to add new tools within the development process.
- Compliancy at CMMI® levels 2 and 3 through the definition of defined and managed standard software processes. The use of method engineering opens the door to the definition and management of the development processes. That is why EzForge will give support, for example, to the use of CMMI®. This will make it possible to ensure that carried out developments will place the organization in a certain maturity level, allowing an improvement of the methodology used
- Establish a relationship between process a project management tools (EPF) and engineering tools (forge).
- Provide a new tool in the knowledge management area through the use of method engineering

Currently we are applying this approach in several test cases and projects. A step forward in this development is to provide facilities to the forge in order to collect all kind of metrics. This characteristic will provide a better understanding of our current developments.

## Acknowledgements

This work was funded by the Vulcano project (Proyecto Singular Ministerio Industria, Turismo y Comercio- FIT-350503-2007-7). We would like to thank Vulcano partners for their great feedback.

## References

[1] McAfee, A.: Enterprise 2.0: The Dawn of Emergent Collaboration. MIT Sloan Management Review 47(3), 21–28 (Spring, 2006)
[2] Soriano, J., Lizcano, D., Cañas, M.A., Reyes, M., Hierro, J.J.: Fostering Innovation in a Mashup-oriented Enterprise 2.0 Collaboration Environment. System and Information Sciences Notes 1(1) (July 2007); ISSN 1753-2310
[3] Huang, L., Boehm, B.: How much Software Quality Investment is Enough: a value-based approach. IEEE Software 23(5), 88–95 (2006); Digital Object Identifier 10.1109/MS.2006.127
[4] Campanella, J.: Principles of Quality Costs. American Society for Quality Press; ISBN: 0-87389-443
[5] Crisis, M.B., Konrad, M., Shrum, S.: CMMI® Second Edition. Guidelines for Process Integration and Product Improvement. Addison-Wesley, Reading; ISBN 0321279670
[6] Booch, G., Brown, A.W.: Collaborative Development Environments. Advances in Computers 59 (2003)
[7] EzForge project website, `http://ezforge.morfeo-project.org/lng/en`
[8] Fielding, R.T.: Architectural styles and the design of network-based software architectures, Ph.D. thesis, University of California, Irvine (2000)
[9] Senge, P.M.: The fifth discipline. Doubleday (1990); ISBN 0-385-26095-4
[10] Brinkkemper, S.: Method engineering: engineering of information systems development methods and tools. Inf. Software Technol. 38(4), 275–280 (1996)
[11] Henderson-Sellers, B., France, R., Georg, G., Reddy, R.: A method engineering approach to developing aspect-oriented modelling processes based on the OPEN process framework. Information and Software Technology, doi:10.1016/j.infsof.2006.08.003
[12] Larrucea, X.: Method Engineering Approach for Interoperable Systems Development. Journal Software Process: Improvement and Practice (2008), doi:10.1002/spip.371
[13] Software Process Engineering Metamodel (SPEM) 2.0. Object Management Group, `http://www.omg.org/docs/ptc/07-11-01.pdf`
[14] Driver, E., et al.: Road Map to an Enterprise Collaboration Strategy. Forrester Research, August 2 (2004)

# Using MDE to Build a Schizophrenic Middleware for Home/Building Automation

Grégory Nain, Erwan Daubert, Olivier Barais, and Jean-Marc Jézéquel

IRISA/INRIA/university of Rennes1, Equipe Triskell, F-35042 Rennes Cedex

**Abstract.** In the personal or corporate spheres, the home/office of tomorrow is soon to be the home/office of today, with a plethora of networked devices embedded in appliances, such as mobile phones, televisions, thermostats, and lamps, making it possible to automate and remotely control many basic household functions with a high degree of accuracy. In this domain, technological standardization is still in its infancy, or remains fragmented. The different functionalities of the various appliances, as well as market factors, imply that the devices that control them communicate via a multitude of different protocols (KNX, LonWorks, InOne). Building a high level middleware to support all the appliances seems to be a reasonable approach. However, market factors has shown that the emergence of a unique and universal middleware is a dream. To solve this issue, we have built a new generation of schizophrenic middleware in which service access can be generated from an abstract services description. EnTiMid, our implementation of schizophrenic middleware, supports various services access models (several personalities): SOAP (Simple Object Access Protocol), UPnP and DPWS (Device Profile for WebServices). In this paper, we describe how these personalities are generated using a Model Driven Engineering approach and discuss the benefits of our approach in the context of a deployment of new services at the city level.

## 1 Introduction

Time after time, each building parts manufacturer has developed his own communication protocol, and this for two reasons. The first one is the increasing need of communication between the devices. Then, the belief that a close protocol is more secured, is still present in minds and make the second reason. As a consequence, devices of today are communicating through dozens of protocols, and most of them are private and protected. For example, X2D[1], InOne[2] or IO-homecontrol[3] are private protocols. Open protocols are emerging such as KNX, LonWorks or BacNet, but interconnections between each other and/or with private protocols are often made 'on demand'. Building a high level middleware to support all the appliances and allow the development of high level services seems

[1] Dela-Dore protocol (http://www.deltadore.com)

[2] Legrand protocol (http://www.legrand.fr)

[3] IO-homecontrol consortium protocol (http://www.io-homecontrol.com)

© Springer-Verlag Berlin Heidelberg 2008

to be a reasonable approach. However, building automation market factors has shown that the emergence of a unique and universal middleware is a dream. To solve this issue, we present in this paper a new generation of schizophrenic middleware [15] in which service access can be generated from an abstract services description. EnTiMid, our schizophrenic middleware implementation, supports various services access models (several personalities): SOAP (Simple Object Access Protocol), UPnP [12] and DPWS (Device Profile for WebServices) [6]. In this paper, we describe how these personalities are generated using an Model Driven Engineering approach and discuss the benefits of our approach in the context of a deployment of new services at the city level.

The remainder of the paper is organized as follows. Section 2 presents an overview of EnTiMid, a middleware for home automation and presents the metamodel embedded into EnTiMid to export devices services at the business level. Section 3 presents the generative tool chain of personalities. Section 4 discusses the usage of EnTiMid in the context of a city deployment. Section 5 highlights some selected related works and Section 6 wraps up with conclusions and outlines some future work.

## 2 Overview of EnTiMid

EnTiMid is a middleware implementation developed in a house or building automation context. The aim of this middleware, is to offer a level-sufficient abstraction, making it possible for high level services to interact with physical devices (such as lamps, heater or temperature sensors).

### 2.1 A Layered Middleware Based on Services

Based on a service-oriented architecture [5], this middleware incites people to build their software as a set of services working together. Thus, each user can customize the services offered by the software, according to his needs.

The OSGi Alliance[9], 'consortium of technology innovators', has released a set of specifications which define a service-oriented platform[1], and its common services. The OSGi kernel is a standard container-provider to built service-oriented software. It implements a cooperative model where applications can dynamically discover and use services provided by other applications running inside the same kernel. It provides a continuous computing environment. Applications can be installed, started, stopped, updated and uninstalled without a system restart. It offers a remote management model for applications that can operate unattended or under control of a platform operator. Finally it embeds an extensive security model so that applications can run in a shielded environment. According to these specifications, an application is then divided on several bundles. A bundle is a library component in OSGi terms. It packages services that are logically related. It imports and exports Java packages and offers or requires services. Services are implementations of Java interfaces.

Inspired by the CBSE (Component Based Software Engineering) paradigm [11], each bundle is designed to reach the highest level of independence, giving the

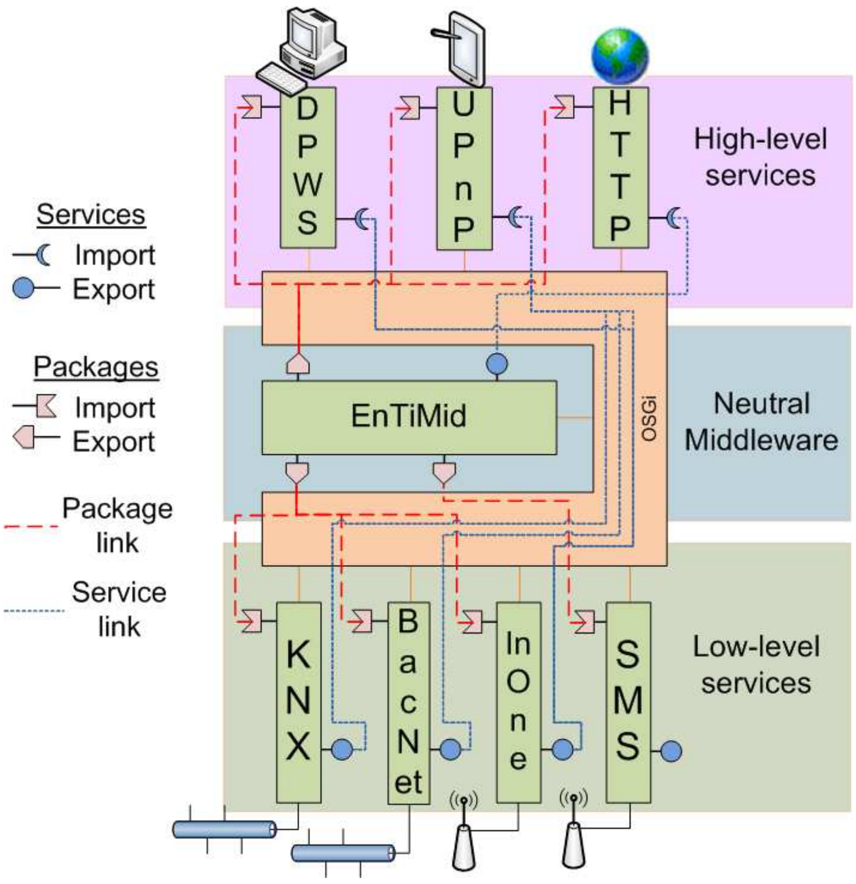


**Fig. 1.** EnTiMid architecture

software enough modularity to allow partial services updates, adds or removes. This programming style allows software-builders, to deploy the same pieces of software for all of their clients, either professionals or private individuals, and then simply adapt the services installed. Moreover, the services running on the system can be changed during execution.

On top of this kernel, EnTiMid is structured around three layers as presented in Figure 1: a low-level layer which manages communication with the physical devices, a middle layer (the kernel), offering a message oriented middleware (MOM), and giving a first abstraction level of low-level layer, a high-level layer which publishes services and enables the device access through standard protocol. The Figure 1 gives an overview on how services are organized. EnTiMid defines a set of interfaces between the services provided by the low level layer and the services required by the middle layer to allow the access to the physical devices.

The low-level offers a common abstraction to the EnTiMid Kernel to access the different devices. It wraps existing library to support some protocols, for example, it wraps the Calimero toolkit to provide an access to KNX devices [4]. It provides also home-made drivers to access protocol such as X2D or X10. The EnTiMid kernel (middle layer) and the high level layer are described more precisely in Sections 2.3 and 3.

## 2.2 A Schizophrenic Middleware

High-level protocols, such as DPWS, SOAP or UPnP, are going to be more and more present in our everyday life. Each of them offers a different access to devices, according to their main goal. For example, UPnP has been developed to ease media sharing, whereas SOAP is a protocol to programmatically access services such as devices provided services. So, manufacturers choosing to implement a high-level access to their devices, will select the protocol offering the best accordance with the devices applications.

In a few years, new high and low level protocols will probably appear, and some will become useless. Existing protocol will evolve and EnTiMid, which aims to ease the device interconnection, has to support all the new protocols and new protocols versions. An interesting solution to face the need of protocol management flexibility (both low and high level), is the schizophrenic middleware, presented by Laurent Pautet[15]. A schizophrenic middleware offers several personalities to access services. Consequently, in EnTiMid, high-level protocols define application personalities. As the middleware has to support lots of application personalities, we propose in this paper to use a MDE approach to generate these application personalities from an internal representation of devices. This choice is driven by the features provided by a MDE approach (abstraction, transformation language) and by the improvement of the maturity of MDE tools (we use the Eclipse Modelling Framework [3] for the meta-model definition and Kermeta [8] for the transformation). Next subsection presents in depth the internal representation of the device services. Section 3 presents the details of the generative approach used for the UPnP and the DPWS personalities.

## 2.3 EnTiMid Kernel

Figure 2 shows a part of the structure of the middle-level layer. Each manufacturer provides a *Bundle*. This bundle will use a *Gateway* contained in a *Zone* in order to access the devices. Each low-level bundle implements a method called *getAvailableProducts()*. This method returns a catalogue of the devices the bundle is able to control/provide. Then, users just have to instantiate the device they need to interact with. According to the type of the device, the actions offered are different.

*Actuators* inform the system about the actions they are able to realise, giving a list of *CommonAction* (on, off, up, down...). Then, for each *CommonAction*, they produce a *HouseAction* containing all necessary information for the action to be done on a device. These *HouseActions* can be valued to specify light intensity for example.

[4] calimero.sourceforge.net

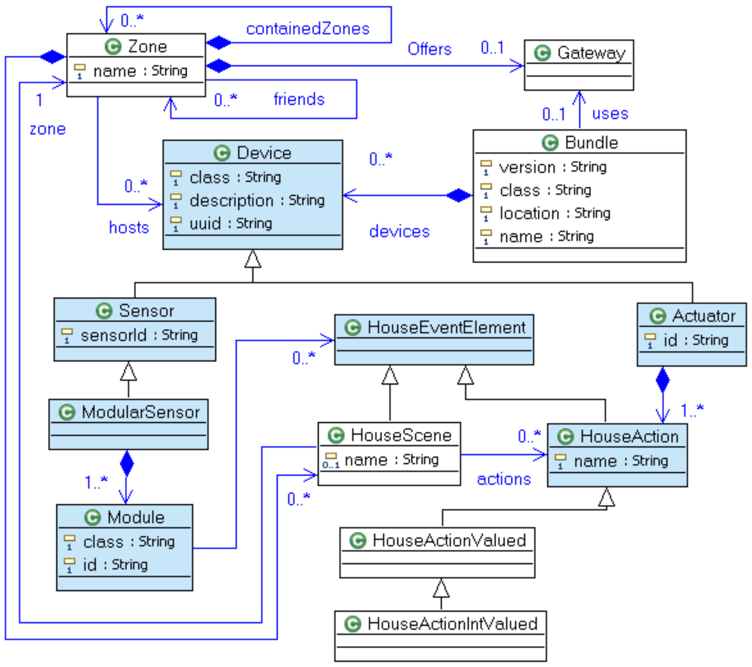


**Fig. 2.** Simple EnTiMid model

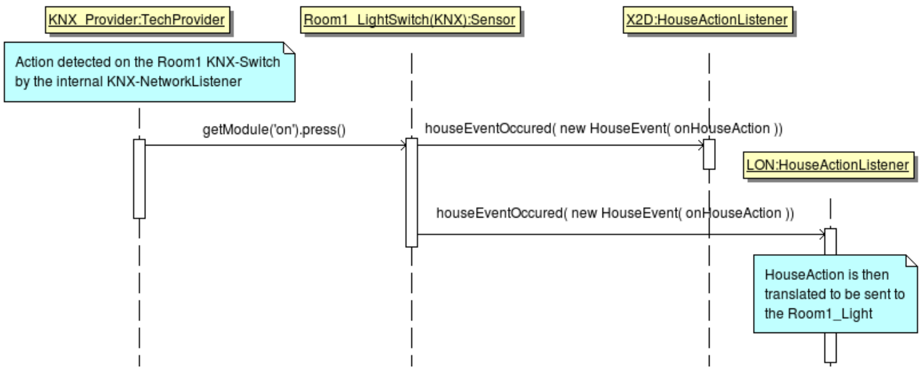

KNX_Provider:TechProvider
Room1_LightSwitch(KNX):Sensor
X2D:HouseActionListener
Action detected on the Room1 KNX-Switch by the internal KNX-NetworkListener
getModule('on').press()
houseEventOccured( new HouseEvent( onHouseAction ))
LON:HouseActionListener
houseEventOccured( new HouseEvent( onHouseAction ))
HouseAction is then translated to be sent to the Room1_Light

**Fig. 3.** Message transmission on action detection

*Sensors* are divided in different categories, and we focus on *ModularSensors*. They are composed of *Modules*, and those modules are only containers. At setup time, a sensor in charge to switch off a light, will ask the light actuator for the *HouseAction* to be sent to switch off the light, and this *HouseAction* is then stored in a list. Then, at use time, when the *push()* method is called on a *Module* of a *ModularSensor*, all the *HouseActions* stored in the *Module* list are published to all *HouseActionListeners*.

Figure 3 illustrates what append when push method is called on a *Module*. A *HouseEvent* containing a *HouseAction* is sent for each *HouseAction* to each *HouseEventListener*. Then, the bundle able to route the information on the destination network (here the LonWorks network) will translate and send the message to the concerned device.

## 3 A MDE Approach to Generate Middleware Personality

### 3.1 From PIM to PSM

The Object Management Group (OMG) promotes an approach to achieve adaptability at the application level with the Model-Driven Architecture (MDA) [10,4]. The concepts involved in MDA are based on the definition of two kinds of model: Platform Independent Models (PIM) and Platform Specific Models (PSM). A PIM provides an abstract representation of an application independently of any standards and aims at capturing only the business concerns. A PSM is the result of the transformation of a PIM to apply it on a particular technology or a particular standard. PIM and PSM are defined by specific meta-models which constrain the expression of the models. The EnTiMid internal device services model can be considered as a PIM, due to the fact that all devices types are specified in the framework, and specifications are common to all technologies. On the opposite, UPnP or DPWS personalities, for example, are to be considered as PSMs, because the device description mode is very specific for each device and dependent of the protocol.

### 3.2 UPnP Personality Generation

**UPnP Meta-model.** UPnP [12] is based on a discovery-search mechanism. As an UPnP-Device joins the UPnP network, it sends an XML description file to all UPnP-ControlPoints, presenting itself with information such as manufacturer, device type, device model or uuid.

Most of times a UPnP-Device is self-contained, describing itself and publishing all services it can offer. The UPnP specification allows devices to contain other devices (called embedded devices); if so, the *rootDevice* (the container) have to publish its self-description and the description of each embedded device.

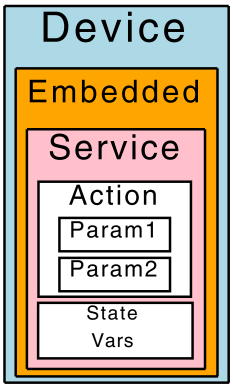


**Fig. 4.** UPnP-Device structure

Moreover, UPnP-Devices (embedded or not) can offer UPnP-Services, as shown in Figure 4. Each service type provided by an UPnP-Device, must be described in a separated file. The description explicits all the UPnP-Actions the service renders, and all the UPnP-StateVariables, used by these actions, in

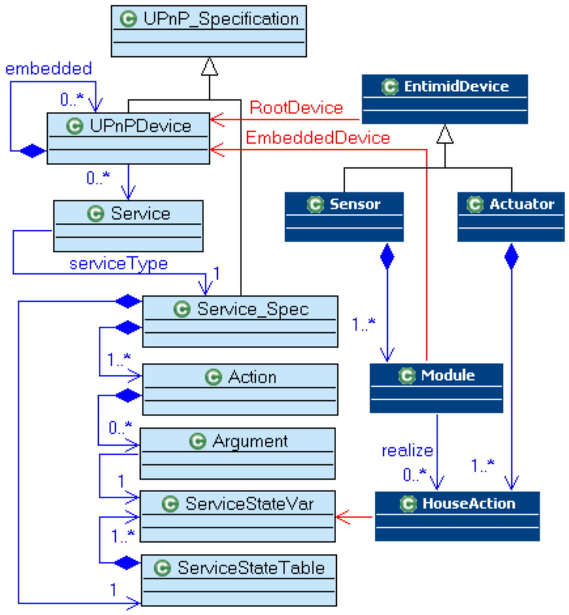


**Fig. 5.** UPnP mapping example

a UPnP-StateTable. Indeed, UPnP-Actions can be parametered, and sens (in or out), name and related StateVar are specified for each parameter. UPnP-StateVars are used to offer more precise information about parameters, such as value type or allowed value list.

**Meta-binding.** The Figure 5 shows how the binding between EnTiMid-Devices and UPnP-Devices is done. An EnTiMid-Device is exported as a UPnP-RootDevice, beeing Sensor or Actuator.

In the case the device is a Sensor, and more precisely, a ModularSensor, all the modules it contains are exported as UPnP-EmbeddedDevices. The RootDevice do not offer any services: only modules are offering such a thing. But, actuators do not contain any Module, and so, can not be exported as simply as sensors.

In order to allow users to generate actions on actuators, a new UPnP-Device is created and is given as *BasicModule* as it offers *CommonActions*. For example, if a simple light actuator offers "on" and "off" *CommonAction*, a *BasicModule* will be created for each action, and added to the RootDevice. By this way, it will be presented as a UPnP-Device containing two Modules (one "on" ans one "off") each one offering a *push()* method.

As a consequence, EnTiMid-Devices are mapped to UPnP-RootDevices; the Modules are mapped to UPnP-EmbeddedDevices and *CommonActions* are specified as *ServiceStateVariables*.

**Files.** As previously said, each UPnP-RootDevice and each UPnP-Service must be defined in an XML description file. During its life, the UPnP-Device will frequently be asked to send its description file to other UPnP-Devices, and services descriptions are consulted each time a service is likely to be used.

In a commercial point of view, those description files are defined once (for a proper device), and embedded in the product. But the dynamics of EnTiMid, its abstraction level and its modularity, implies that information, such as device type or description, about the devices installed on the system, are never known in advance. Moreover, the EnTiMid implementation of a product can offer different services, and those services are implementation dependent. That is why the description files are generated at runtime.

**Service description file generation.** Our choices of implementation has led to the fact that only Modules offer services. UPnP-Actions, offered by a UPnP-Service, are an UPnP-Export of some methods of the module class. Methods to be exported are signaled to be UPnP-Compliant, thanks to the presence of an UPnP-Method annotation. This annotation offers some interesting information. The first information is that the annotated method has to be exported in the service; then, the annotation gives, at runtime, a semantic name to the parameters and the returned value of an action. More precisely, without annotation, the only information one can get on a parameter at runtime is its place on the method signature and its type. In this case, the user can not have the information that the first string parameter is for the name, and the second for the age. The annotation brings these information.

So, if the BasicModule service description file does not exist yet, a simple reflexive research on the class's methods make it possible for the system to generate it, and all devices offering this service will then be able to point to this file.

**Device description file generation.** In order to generate the description file of a device only once, the name of file is the device class name. Device des s. The first one gives general information about the device: manufacturer, description, model type, model number or uuid; some are statically completed by the EnTiMid-System (such as manufacturer), others are retrieved from the device itself (model type, model number). The second part contains the types, identifiers and links of the services offered by the device. For example, the first BasicModule declared will complete its services list with:

```
<serviceType>urn:www.entimid.org:service:BasicModule:1</serviceType>
<serviceId>urn:www.entimid.org:serviceId:BasicModule1</serviceId>
<SCPDURL>/service/BasicModule/description.xml</SCPDURL>
<controlURL>/service/BasicModule/control</controlURL>
<eventSubURL>/service/BasicModule/eventSub</eventSubURL>
```

The last description file part contains the description of each embedded devices. Those descriptions are composed of the two previous part, for each embedded device.

**UPnP events management.** A different listener is created for each service of the device, to simply manage UPnP events. By this way, a listener is linked with a unique module, and side-effects with other modules or devices are avoided. Each listener is then attached to each action of the service.

When an action event is received, the first work is to identify the method that has been actionned. Once done, the second work is to cast the UPnP-Action parameters into the real method argument types. Then the method is invoked, and, if necessary, the result is translated into a UPnP-VarType and sent.

### 3.3 DPWS Personality Generation

The Device Profile for Web Services (DPWS) [6] defines a minimal set of implementation constraints to enable secure Web Service messaging, discovery, description, and eventing on resource-constrained devices. Its objectives are similar to those of Universal Plug and Play (UPnP) but, in addition, DPWS is fully aligned with Web Services technology. Moreover, it includes numerous extension points, allowing for seamless integration of device-provided services in enterprise-wide application scenarios. From a conceptual point of view, the DPWS meta-model is closed to the UPnP meta-model described in Figure 4. Consequently, building the abstract model of the service to export, follows the same way: we use the Java annotation in the low level layer to infer the model. However, the generation process is different. To build the DPWS layer, we use the WS4D project [16]. This project proposes a programming model to create DPWS services. This model is based on the concept of *service*, *device*, *operation* and *parameter*.

A DPWS *service* provides an implementation of one or more WS (Web Services) port types to DPWS clients. The messages, a service receives and sends, are specified by its WS port types. The DPWS services definition is different of the standard definition of the term *service* in the WSDL specification. A *device* hosts one or more services and provides common functions like messaging or discovery. It is classified by its port types. According to the DPWS specification a device is a target service of the WS-Discovery specification. The basic functionality of a device is implemented by the class HostingService. An *Operation/Action* is a named message exchange pattern of a port type. It can consist of an input and output message, and multiple fault messages. The appearance and order of the input and output messages in the WSDL determine the type of the operation (request-response (in-out), solicit-response (out-in), notification (out), one-way (in)).

For each device, service and operation, a Java class has to be generated. This class must extend respectively HostedService, HostingService and Action. For each parameter, an instance of the class Parameter has to be implemented. Consequently, the generation process can be done automatically. To achieve that we use the JET Framework to create generation template for DPWS. We embed the JDT compiler provided by the eclipse project to compile the generated code. Finally, we programmatically create a new bundle containing all generated classes. Once loaded, this new bundle provides all generated classes, and allow them to be used.

## 4 Use Case

### 4.1 Context: Application to a City-Level Project

EnTiMid is currently used in a Brittany project to allow old persons to hold in their home as long as possible. Two associations of the Rennes metropolis, the "*CODESPAR*" and the "*ASSAD*", are working together in a project called "*Maintiens à domicile et habitat évolutif*". With the support of industrial partners, they are conceiving an environment around health professionals and old people, composed of new information technologies. As previously said, the main goal of the project is to help people to stay at home as long as possible, but this can not be done without helping health professionals in their everyday work.

From March to October 2007, an initial study has permitted to obtain a set of recommendations. The second phase of the project aims to find technical answers to these recommendations. However, technical solutions are often multiples, and the probability to install this technical environment over, or mixed with, an already installed technologies is not null.

Consequently, the software used to manage the technologies and ease the access to the house for health professionals, will have to be fully adaptable to the in-place technology, and require a short development time to reach new technologies or new protocol versions.

Its unified technology management, provided by the middleware abstraction of the underlying protocols, and its multiple access personalities, inherited from its schizophrenic aspect, have led EnTiMid to be a privileged candidate to be deployed as the access point to the equipped houses.

### 4.2 Advantages of a Schizophrenic Middleware in This Context

The schizophrenia of this middleware and its generative capabilities are advantages in two dimensions.

**In space.** The city scale deployment of the project necessarily implies that the technologies used will sometimes be different, due to some physical constraints, or because a technology is already installed, and people do not want to change. EnTiMid will then propose an abstraction of the deployed devices technologies; it will expose different personalities of these devices for high level application developers. Consequently, services provider associated to the project will be able to develop high level services directly on top of DPWS or UPnP. Finally, the management capabilities provided by the OSGI gateway will also help to update and reconfigure the gateway.

**In time.** Software, technologies and protocols constantly evolves and versions change with, sometimes, some compatibilities problems. That is to say that during its life, the OSGi gateway will have to implement new protocols, or different versions of a given protocol. Moreover, protocols can be used in different versions, at a given time, in different places of the city. Once again, the different personalities make it possible for EnTiMid to gain multiple version compliance, for different protocols.

## 5 Related Work

Xie Li [7] has developed a residential service gateway, which aim is to rely "inside-of-house" system to a "outside-of-house" system, allowing users to connect their residential gateway from a centralized point "outside-of-house". The connection is recommended to be done by a VPN solution, and offers an HTTP interface to control devices through the Lonworks PLC technology and they have planned in future work to export their services to UPnP. The "inside-of-house" system, developed on an OSGi platform, implements algorithm giving Plug&Play facilities to the system for "pre-defined devices". Compared to this system, our implementation is designed to be Plug&Play. Our system also eases the interoperability and can access several technologies.

The paper [13] presents a "Service Oriented Pervasive Applications Based On Interoperable Middleware". As EnTiMid, the described middleware is composed of three layers: a drivers layer, in charge of the connexion between the devices and the "Unified Service" layer. Then a bridges layer links the Unified Service instances to diverse "service technologies (UPnP, WS,...)". The solution is similar, but they do not use the OSGi technology. Besides, we made the choice in this paper to use an MDE based generative approach to propose several personalities for the diverse "service technologies (UPnP, WS,...)".

Valtchev *et al.*[14] have developed a gateway to control a smart house. This gateway defines an abstract layer to manage the hardware protocol used to communicate with physical devices. Moreover it is defined to manage many services gateways. But it does not define a high level abstraction to offer services through protocols like DPWS or UPnP. This abstraction could be done using their gateway but for practical reason we have choosen to define our own implementation, because their implementation offers many things like we don't want to use now. Even if, later, EnTiMid could be bigger and need to manage many gateways for examples.

Bottaro *et al.* have developed a service platform[2] to offer service abstraction like DPWS over device communication. Into this platform, each device has to register on the OSGi context, for each high level protocol it implements. At runtime, the high level protocol 'manager' gets all registered devices and publishes them on the network. The main difference between this platform and EnTiMid comes from the service registration. Indeed, for each high level protocol a device want to offer, it has to implement a set of specific interfaces(API) and register to the OSGi context. The generative approach used in EnTiMid simplifies the development of devices. All installed devices are natively exported toward high level protocols, thanks to their EnTiMid-Device implementation.

## 6 Conclusions and Perspectives

The plethora of networked devices embedded in appliances, such as mobile phones, televisions, thermostats, and lamps, makes possible to automate and remotely control many basic household functions with a high degree of accuracy.

Consequently, a new breed of technologies is needed to address the challenges of the development and deployment of building automation applications over an evolving, large-scale distributed computing infrastructure. The approach and the tools, provided by EnTiMid, and described in this paper, are an example of such a technology.

EnTiMid offers a first solution to manage the multiplicity and the evolution of communication protocols through a layered schizophrenic middleware. This solution consists of offering a common abstraction of the home device topology and provides a generative approach to offer an access to the devices through different personalities. To improve the flexibility of this middleware, the high level protocols are generated and loaded at runtime. It enables a dynamic reconfiguration of the application and the high-level protocol bundle without any system restart.

EnTiMid have been implemented to form a complete middleware for home automation[5]. Future work includes technical improvement and new scientific investigations. As a technical improvement of the platform, the AMIGO European project designed a 3D application called VantagePoint, in order to model a room with objects and devices. Moreover, the JDT compiler embedded to compile the DPWS personalities is heavyweight for implementation in small commodity set-top box. The DPWS generation tool chain has to be technically improved. This could be a really good way to generate the configuration file of a house, or to provide a 3D device management application. As a scientific future work, we will work on the definition of a context-aware service composition operator in order to provide users the relevant high level services. In this context, we will follow the work of the S-Cube project in particular the work on the adaptation of service-oriented applications.

**Acknowledgments.** 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). (http://www.s-cube-network.eu).

## References

1. The OSGi Alliance. Osgi service platform core specification, release 4, avril (2007)
2. Seyvoz, S., Bottaro, A., Simon, E., Gérodolle, A.: Dynamic web services on a home service platform. In: 22nd International Conference on Advanced Information Networking and Applications, pp. 378–385 (March 2008)
3. ECore. The eclipse modeling framework project home page, `http://www.eclipse.org/emf`
4. Fuentes, L., Pinto, M., Vallecillo, A.: How mda can help designing component- and aspect-based applications. In: EDOC 2003: Proceedings of the 7th International Conference on Enterprise Distributed Object Computing, Washington, DC, USA, p. 124. IEEE Computer Society, Los Alamitos (2003)

[5] http://house-manager.gforge.inria.fr

5. Jammes, F., Smit, H.: Service-oriented paradigms in industrial automation. IEEE Trans. Industrial Informatics 1(1), 62–70 (2005)
6. Jammes, F., Mensch, A., Smit, H.: Service-oriented device communications using the devices profile for web services. In: MPAC 2005: Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing, pp. 1–8. ACM, New York (2005)
7. Li, X., Zhang, W.: The design and implementation of home network system using osgi compliant middleware. IEEE Transactions on Consumer Electronics 50 (May 2004)
8. Muller, P.-A., Fleurey, F., Jézéquel, J.-M.: Weaving executability into object-oriented meta-languages. In: Kent, S., Briand, L. (eds.) MoDELS 2005. LNCS, vol. 3713. Springer, Heidelberg (2005)
9. Osgi alliance, http://www.osgi.org/About/HomePage
10. Soley, R., OMG Staff: Model-Driven Architecture. OMG Document (November 2000)
11. Szyperski, C.: Component technology: what, where, and how? In: ICSE 2003: Proceedings of the 25th International Conference on Software Engineering, Washington, DC, USA, pp. 684–693. IEEE Computer Society, Los Alamitos (2003)
12. The UPnP Forum, http://www.upnp.org
13. Uribarren, A., Parra, J., Uribe, J.P., Makibar, K., Olalde, I., Herrasti, N.: Service oriented pervasive applications based on interoperable middleware. In: Workshop on Requirements and Solutions for Pervasive Software Infrastructure (RSPSI 2006) (2006)
14. Valtchev, D., ProSyst Software AG, Frankov, I.: Service gateway architecture for a smart home. IEEE Communications Magazine 40, 126–132 (2002)
15. Vergnaud, T., Hugues, J., Pautet, L., Kordon, F.: Polyorb: A schizophrenic middleware to build versatile reliable distributed applications. In: Llamosí, A., Strohmeier, A. (eds.) Ada-Europe 2004. LNCS, vol. 3063, pp. 106–119. Springer, Heidelberg (2004)
16. Zeeb, E., Bobek, A., Bohn, H., Prueter, S., Pohl, A., Krumm, H., Lück, I., Golatowski, F., Timmermann, D.: Ws4d: Soa-toolkits making embedded systems ready for web services. In: 3rd International Conference on Open Source Systems, Embedded Workshop on Open Source Software and Product Lines, Limerick, Ireland (2007)

# Model-Driven Integration and Management of Data Access Objects in Process-Driven SOAs

Christine Mayr, Uwe Zdun, and Schahram Dustdar

Distributed Systems Group
Information Systems Institute
Vienna University of Technology, Austria
christine.mayr@inode.at, {zdun,dustdar}@infosys.tuwien.ac.at

**Abstract.** In most process-driven and service oriented architectures (SOA), services need to access data stored in a database using database transactions. This is typically done using Data Access Objects (DAOs), but so far the integration of the business process, service, and DAO concepts is not well defined. As a consequence, when the number of services in a SOA grows, the number of DAOs can increase considerably and become hard to manage. In addition to this technical issue, business processes have to be highly adaptable to both functional and technical requirements. We propose a model-driven approach for integrating and managing DAOs in process-driven SOAs. We present a set of models providing different views tailored to the requirements of various stakeholders, such as business experts, database designers, database developers, etc. In process-driven SOAs, process activities running in a process-engine invoke services. We adapt these process flows to model a sequence of DAOs within a service. A DAO repository is used to manage DAOs more efficiently and to improve software reuse in database transaction development. The repository provides functionalities to create, update, retrieve, and delete database transactions. As our view-based models can flexibly be adapted to project requirements, our approach also aims at enhancing maintainability and increasing software development productivity.

## 1 Introduction

In a process-driven, service-oriented architecture, services are invoked from process activities running in a process engine [1]. In this paper we concentrate on an important part of process-driven SOAs: persisting the business objects (and other data) that is used and manipulated by the processes and services. Nowadays, this is often done by integrating Data Access Objects (DAOs) into services. DAOs are a special kind of objects providing access to data that is usually read or written from one or more database tables. Services invoke the DAOs to commit a database transaction to persistent storage. The goal of this design is to enhance software maintainability and strict separation of the layers providing business functionality and data access in a SOA. In addition, DAOs provide an interface that is independent of the underlying database technology. Common DAO implementations are provided by object-relational mapping (ORM) tools, such as Hibernate [2] or Ibatis [3]. ORM frameworks support developers in mapping data between object-oriented programming languages and relational database management systems (RDBMS).

© Springer-Verlag Berlin Heidelberg 2008

As the number of DAOs, as well as the number of uses of a DAO, grows along with the number of services, maintaining and managing the DAOs becomes increasingly difficult when the SOA grows. That is, it becomes hard to deal with tasks such as finding out which services or processes require which DAO, deciding whether a DAO can be discarded because it is not used anymore, or whether a suitable DAO for a specific task has already been defined and can be reused. Different stakeholders involved in a process should be able to understand the SOA only from their perspective. For instance, data analysts require mainly information about which DAOs access which data, service developers require DAOs rather as interfaces to the data, and software architects require the big picture of service/DAO interconnection.

DAOs are an integral part of many SOAs, but unfortunately services and DAOs are not well integrated yet. A straightforward approach to solve this problem are cartridges, such as those provided by AndroMDA [4] or Fornax [5]. Cartridges support separation of concerns by providing mechanisms for accessing and manipulating data through DAOs. They are predefined components for model-driven generators that enable developers to integrate DAOs into services by generating either an instance of a DAO interface into the service code [4] or generating DAO instances into the service operation [5]. However, the relationships between DAO operations and service operations are not specified so far by cartridges. Even though the Fornax cartridge [5] connects DAOs to service operations, it lacks information about which DAO operations are invoked by which service operation. This information, however, is important for stakeholders, such as software architects and service developers, to gain a clear view about which database transactions are invoked by which service operation. To overcome this problem, we extend the cartridge approach with the integration of DAO operations into service operations and the introduction of service operation flows consisting of DAO operations.

In our earlier work, we introduced a view-based modeling framework (VbMF) [6] for separating different concerns in a business process into different views.The idea is to enable stakeholders to understand each view on its own, without having to look at other concerns, and thereby reduce the development complexity. VbMF is based on model-driven software development (MDSD) [7], i.e., executable code, such as BPEL/WSDL in the example, can be generated from the views.

In this paper, we tackle the problems of integrating DAOs into a process-driven SOA by extending the VbMF using different *views* for managing data objects more effectively and efficiently. One important aspect of our solution is the need for a specific view: the Data Access Object Repository View that particularly offers a fast and efficient retrieval and management of DAOs. Establishing a Data Access Object Repository increases both software development productivity and maintainability by enabling loose coupling between DAOs and services, but still supporting the management of DAOs.

This paper is organized as follows: First, we present our approach for managing and integrating DAOs into process-driven SOAs from the architectural integration point of view. Section 3 provides a specification of the view-based data modeling framework and discusses the model-driven aspects of our solution. We validated the applicability of our approach through a case study in the area of modeling jurisdictional provisions in the context of a district court, described in Section 4. Section 5 discusses related work and finally Section 6 concludes.

## 2 Architectural overview

In this section we propose an approach for integrating and managing DAOs in process-driven SOAs. We discuss the relevant components and the relationships between them from an architectural point of view. In the next section we explain how to support this architecture using our view-based models. The goal of our integration and management approach is to enable effective software development, extended analysis methods, and efficient management of data-related issues in a process-driven SOA. As shown in Figure 1 the architecture consists of four main components:

1. *Process Flow:* An executable Process Flow, such as the BPEL process flow in the example configuration in Figure 1, consists of process activities that each can invoke a service operation. Process activities act as service requesters that invoke a specific service from a service repository.
2. *Service Repository:* The Service Repository serves providers for publishing and requesters for retrieving and invoking services. When a requester discovers a suitable service, it connects to the specific service provider and invokes the service [8]. In VbMF, the service repository is modeled by the Collaboration View. It supports creating, updating, retrieving and deleting of services.
3. *Service Operation Flow:* When a service operation invokes one or more DAO operations, the Service Operation Flow orchestrates these DAO operation calls by a data flow of basic process elements such as sequences, flows, and switches.
4. *DAO Repository:* The Data Access Object Repository is used as central component for managing DAOs. It provides basic functionality for accessing DAOs and in particular aims at efficiently retrieving a suitable DAO operation.

These components are part of the runtime architecture generated from our models. In MDSD [7] models are used to abstract from programming languages and platform-specific details. In this paper, we use BPEL as an example for a specific process execution language. In the following sections, we focus on the Service Operation Flow and the DAO Repository that are novel contributions of the work presented in this paper.

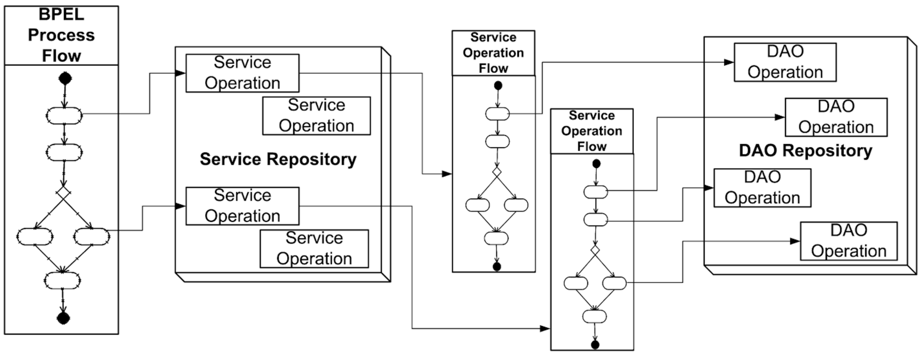


**Fig. 1.** DAO Service Integration Overview

## 2.1 DAO Repository

As the number of DAOs can grow considerably large as the number of services grows, retrieving a certain DAO operation, for instance for reusing the operation, can be complex and time consuming. Also, the potentially large number of DAOs must be managed. For instance, it must be decided which DAO is obsolete and which DAOs can be merged. To support these issues, in our integration and management architecture, we established a DAO Repository as a central component for managing DAOs efficiently. Our DAO repository interface provides the following basic functionality:

– *Insert:* Create a new DAO operation
– *Update:* Change operation name or parameter definitions of a DAO operation
– *Search:* Retrieve a DAO operation by certain search criteria
– *Delete:* Remove a DAO operation

The DAO Repository is the central component for publishing and discovering DAO operations – the functional parts of a DAO. Within the DAO repository each DAO operation belongs to one or more database tables and in the end to a database. This classification enables advanced searching capabilities. Listing 1.1 shows a few example queries for selecting DAO operations by different search criteria. The query `getDaoOperation` returns a list of DAO operations matching a pattern describing a DAO operation name. The second query operation `getDaoOperationByDAO` returns a list of DAO operations belonging to the given `daoObject` type. The third query `getDaoOperationByDbName` returns a list of DAO operations by a given pattern describing the name of a certain database instance. Finally, the query `getDaoOperationByTableName` returns a list of DAO operations accessing a certain database table. In Section 4 we apply the DAO repository using a concrete example.

```
DaoOpList[] getDaoOperation(String pattern)
DaoOpList[] getDaoOperationByDAO(DAOObject daoObject)
DaoOpList[] getDaoOperationByDbName(String pattern)
DaoOpList[] getDaoOperationByTableName(String dbPattern, String tablePattern)
```

**Listing 1.1.** DAO Repository: Query examples

## 2.2 Service Operation Flows

In this section, we describe the Service Operation Flows (see Figure 1) in more detail. A Service Operation Flow supports separation of concerns by enabling service developers to extract the database transactions from the service implementation. This way, service developers get a clearer understanding about the data flows within a service. As a result, development complexity decreases and higher-quality documentation can be generated.

To specify Service Operation Flows we have to integrate DAO operations into service operations. Cartridges, such as AndroMDA [4] and Fornax [5], relate DAOs to services rather than service operations to DAO operations. Establishing the relationships between service operations and DAO operations provides the basis to enable advanced analysis capabilities of database transactions in process-driven SOAs. For example, we can estimate the average number of calls of a specific DAO operation within a process flow for tuning technical database parameters or database indexes. Database indexes can significantly improve the performance of database transactions, but it must be considered that update costs have a considerable influence, both in the context of the physical database design and in access path selection for RDBMS query optimization [9]. The specification of a database transaction provides the information about which database indexes are required for a specific application. However, database transactions that are never invoked must not be selected for index creation.

A DAO consists of a set of DAO operations, and a service consists of `0..n` service operations. When a service operation needs access to the database, it invokes a DAO operation from the DAO repository. However, often a service operation invokes more than one DAO operations that can be dependent on each other. Therefore we introduce Service Operation Flows consisting of database transactions (DAO operations). Our Service Operation Flows support at the moment sequence and switch elements, but can be extended with any other kind of data flow structure. We show a detailed example of a Service Operation Flow in Section 4.

## 3 Model-Driven Integration of DAOs into Process-Driven SOAs

In this section we present a model-driven solution for the proposed integration and management architecture of DAOs in process-driven SOAs presented in Section 2. For that purpose we introduce the View-based Data Modeling Framework (VbDMF) as an extension of the Viewbased Modeling Framework (VbMF) described in detail in [6]. The rectangles in Figure 2 display models of VbMF and the ellipsoidal boxes denote the additional models of VbDMF. In VbMF new architectural views can be *designed*, existing models can be *extended* by adding new features, views can be *integrated* in order to produce a richer view, and using *transformations* platform-specific code can be generated. As displayed by the dashed lines in Figure 2 the new view models of VbDMF extend basic VbMF views namely the Information View, the Collaboration View, and the Control-Flow View. The dotted lines in Figure 2 are used to display view integration, e.g., the Service Repository View integrates the Collaboration and Information View to produce a combined view.

We describe the resulting view models specifying the architectural components in Figure 1. Following VbMF's concepts, we distinguish low-level, technical views from

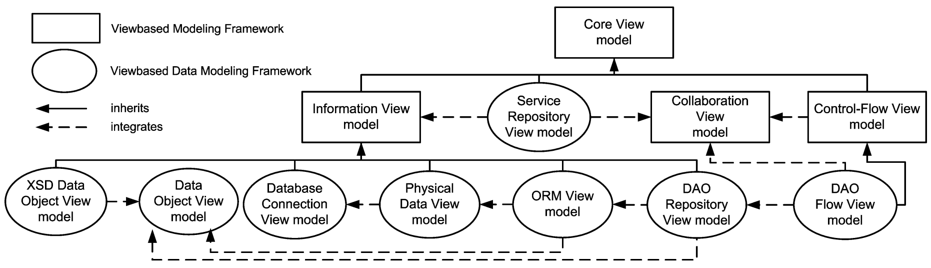


**Fig. 2.** VbDMF and VbMF – Overview

high-level, conceptual (i.e., often: business-oriented) views. In addition, our low-level technical view models support separating technology-specific from technology-independent views, both for presenting the information in the models to different stakeholders and for supporting platform-independence via model-driven software development.

## 3.1 Control-Flow View (BPEL Process) – High Level

The Control-Flow View is an essential part of the Viewbased Modeling Framework [6]. It extends the Core View model (see Figure 2) that defines basic elements of the framework. VbMF introduces this Core View model that is derived from the Ecore meta-model [10] to define and maintain relationships with other view models. The Control-Flow View describes the control-flow of a process, so we can for instance apply it for specifying the BPEL Process depicted in Figure 1. As shown in the figure, a BPEL Process activity can invoke a service operation from the Service Repository to realize the task of the activity. The relationship between a process-activity and a service operation is modeled by the Control-Flow View model by view integration with the Collaboration View model.

## 3.2 Service Repository View (Service Repository)– High Level

The Service Repository View integrates the Collaboration View and the Information View. Both view models belong to VbMF [6] and are derived from the Core View model (see Figure 2). The Collaboration View basically defines service operations required by a process activity. It defines the information needed to generate a Web Service Description Language document (WSDL). The Information View specifies the service operations in more detail by defining data types and messages. Technically speaking, the data types of the Information View are used to generate a schema definition (XSD). In distributed systems, data is passed from one system to another. Each system has its own underlying data structures. For this purpose we specify data type mappings to support data interoperability between diverse systems:

*XSD Data Object View.* The XSD Data Object View specifies conversions between Web Service Description Language (WSDL) Schema types and data types of the service providers' software system environment. For this purpose a class `XsdObjectMapping` associates each `XsdAttribute` with an `Attribute` of a locally defined Data Object.

## 3.3 DAO Flow View (Service Operation Flow)– High Level

The DAO Flow View model extends the Control-Flow View model in order to specify Service Operation Flows, as illustrated in Figure 1, and integrates the Collaboration View model to associate each flow with a specific service operation. The primary entity of the Control-Flow View is the `Activity` class that is the base class for other classes such as `Sequence`, `Flow`, and `Switch` [6]. We extend the class `Activity` to associate a service operation `Operation` of the Collaboration View with a flow of DAO operations. The Control-Flow View model uses the class `SimpleActivity` for representing a concrete process activity [6]. By extending the `SimpleActivity` class of the Control-Flow model we can associate each activity `SimpleActivity` with a `DAOOperation`.

## 3.4 DAO Repository View (DAO Repository)– Low Level

The DAO Repository View is a combined view that integrates the Object Relational Mapping (ORM) View model and the Data Object View model. Since the main purpose of the ORM View is to map physical data to data objects, it consists of both the Physical Data View model, integrating the Database Connection View and the Data Object View. As a result of this view integration, a DAO Repository service can process complex queries for retrieving a specific DAO operation by joining the data from different views (see Section 2.1). DAOs provide an interface that is independent of the used specific ORM technology. That is, this view model specifies a conceptual view rather than a technical view. It consists of a list of `DAOOperation` elements that each holds `0..n` `InputParameter` parameters and a `ReturnParameter`.

*Database Connection View.* The Database Connection View comprises a list of arbitrary, user-defined connection properties and therefore is a conceptual rather than a technical view. We also support database driver dependent views through model extension, e.g., a JDBC Database Connection View.

*Physical Data View.* The Physical Data View contains a class `Database Connection Pool` specified by a list of the class `DBConnection`. We reference the class `DBConnection` of the Database Connection View using model integration mechanisms (see Figure 2). The Physical Data View contains two more basic classes: `Tables` and `ColumnTypes`. We support most common data types for current RDBMSs. As data types can differ among different RDBMSs, developers can create a technology-dependent view by extending this conceptual view model.

*Object Relational Mapping View.* The Object Relational Mapping View is a technology-dependent model that provides the basis for specifying object relational mapping mechanisms in VbDMF. The defined elements result from studying a range of ORM tools in particular Ibatis [3] and Hibernate [2]. In order to support ORM framework's special features, developers should specify a technology-dependent view by model extension. The basic view specifies a mapping between the two below-mentioned models – the Data Object View model and the Physical Data View model. The class `DataObjectToTableMapping` maps a data object (`DataObject`) to a database table (`Table`). The class `MemberVariableToColumnMapping` allows for a more specific mapping between `MemberVariable` and a table `Column`.

*Data Object View.* In object-oriented programming languages information is stored in the objects' member variables. We provide a conceptual, technology-independent model, that consists primarily of a list of data objects `DataObject` and types `MemberVariableTypes`. Again, to define additional data types developers can extend this view model to gain a technology-dependent view.

## 4 Case Study

In this section we present a case study in the area of modeling jurisdictional provisions in the context of a district court, which we have performed to validate the applicability of our approach. First of all, let us explain the Business Process flow and an exemplary Service Operation flow (see Figure 3) at the land registry court. We use UML extensions to model our process flows.

As shown in Figure 3, the process starts when a new application is received. The `ValidateApplication` activity invokes a service that checks the incoming jurisdictional application for correct syntax and semantic. Successfully validated applications are saved by the service triggered by the `SaveApplication` activity. The activity `DeliverDismissal` invokes a service that returns incomplete or inconsistent applications back to the applicant. Stored applications can be executed by the registrar within the human process activity `ExecuteApplication`. If the registrar approves the application, the service-based activity `AccountFees` is invoked. Finally, after accounting the fees, the next activity `DeliverApprovalAndFees` calls a service that delivers an approval and, if required, a pre-filled payment form to the applicant. In case of dismissal the activity `DeliverDismissal` invokes a service that informs the applicant about the dismissed application.

Let us now illustrate how to integrate DAO operations into service operations using the process activity `AccountFees` as an example to demonstrate the database transaction flow within an activity. In Figure 3 the dashed lines indicate the data flows between the DAO operations. The DAO operation `isExemptedFromFees` of `ApplicantDAO` checks whether the applicant is exempted from fees. If the DAO operation `isExemptedFromFees` returns `true`, no further operations are necessary

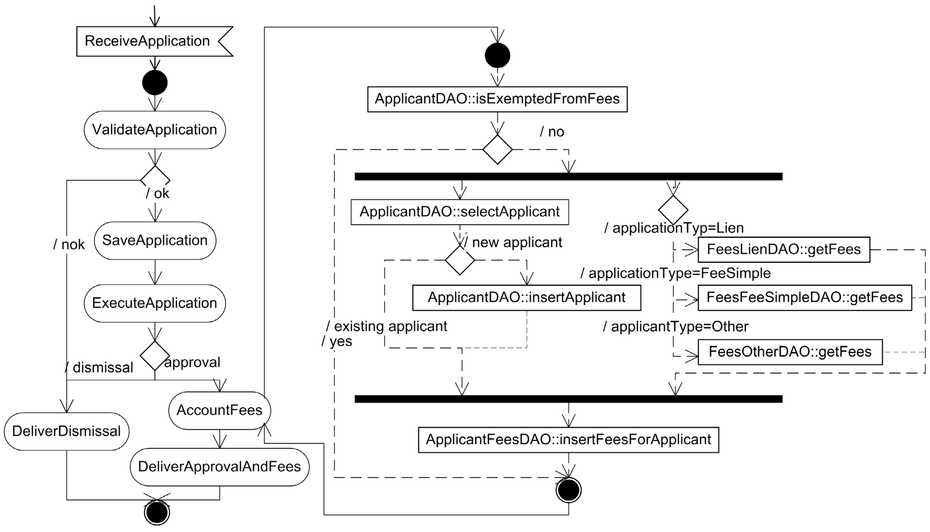


**Fig. 3.** Case Study: Process Flow and Service Operation Flow (Process Activity AccountFees)

**Table 1.** Case Study: DAO Repository View

| DAO | DAO operation | Transaction Type | Database | Table |
|---|---|---|---|---|
| ApplicantDAO | isExemptedFromFees | select | DBTest1 | Applicant |
| ApplicantFeesDAO | insertFeesForApplicant | insert | DBTest1 | ApplicantFees |
| ApplicantDAO | isExemptedFromFees | select | DBProduction1 | Applicant |
| ApplicantDAO | selectApplicant | select | DBProduction1 | Applicant |
| FeesLienDAO | getFees | select | DBProduction2 | FeesLien |

to terminate the flow. Otherwise the DAO operations for accounting fees and selecting the applicant can run in parallel, because their data-flows are independent from each other. Fees are accounted by the fees department and are system-dependent on the type of application: Fees for applications of type 'Lien' are calculated by the DAO operation `getFeesForApplicationType` of DAO `FeesLienDAO`, the DAO operation `getFeesForApplicationType` of DAO `FeesSimpleFeeDAO` accounts fees for applications of type 'FeeSimple', and the DAO operation `getFeesForApplicationType` of DAO `FeesOtherDAO` calculates the costs for applications of type 'Other'. The DAO operation `insertFeesForApplicant` of DAO ApplicantFeesDAO requires a stored applicant as input parameter. For this purpose the `ApplicantDAO`'s DAO operation `selectApplicant` looks for an existing applicant. In case the applicant is not stored in the database yet, the DAO operation `insertApplicant` is invoked to return the new applicant required as an input parameter for the DAO operation `insertFeesForApplicant`. Let us now consider the main view models instances specifying the process flow illustrated before:

*Control-Flow View instance.* In our prototype implementation the Control-Flow View specifies a BPEL Process Flow. A graphical layout of the resulting BPEL model instance is depicted left in Figure 3. The BPEL source code that is not shown here represents another layout of this view instance.

*Service Repository View instance.* The Service Repository View models the services and data types of services often invoked by a process activity. In our concrete example, the Service Repository View instance specifies the services depicted left in Figure 3.

*DAO Repository View instance.* Table 1 shows an extract of the data stored in the DAO Repository after modeling our example process. Due to the view integration mechanisms (Database Connection View, Physical Data View, etc.) this DAO Repository View instance contains data of various categories (DAOs, database, tables, etc.). In our example we query all DAO operations that belong to a certain database 'DBTest1'. In another query we can ask for existing DAO operations depending on a specific table such as 'Applicant'.

*DAO Flow View instance.* The DAO Flow View instance depicted in Figure 3 illustrates the connection between the process activity 'AccountFees' and its Service Operation Flow. In general, the DAO Flow View is intended for software architects, data analysts, and service developers to get both a general understanding of the data flows within service operations and to get the information which service operation depends on which DAO operations. Furthermore we can use this view to query all DAO operations that are needed by a specific process.

We use the oAW's Xpand language for source code generation. A BPEL definition for the process flow and a service description in WSDL are generated from an instance of the Control-Flow View, the Information View, and the Collaboration View respectively. The Apache Axis2 Web services engine [11] supports us in building Java proxies and skeletons for the services with WSDL descriptions. So we can generate a service implementation and use the DAO Flow View model instance to inject the flow of DAO operations into the relevant service operations. The DAOs themselves are generated from plenty of model instances, namely of the DAO Repository View, the ORM View, and the Data Object View. In contrast to generating DAOs, the DAO interfaces are automatically implemented simply from the DAO Repository View instance and the Data Object View instance.

## 5 Related Work

As mentioned before, our approach is related to other model-driven solutions for integrating DAOs into services, such as AndroMDA's EJB3 cartridge [4], generating a persistent tier by integrating DAOs into services, and the Fornax platform [5], aiming at a more specific integration by modeling the relationships between DAOs and service operations. In our solution we associate DAO operations with service operations and thus provide a more in-depth integration solution than these cartridges. Furthermore, in contrast to earlier model-driven approaches, in our approach a data flow of database transactions is modeled within a service operation that can be used to extract data dependencies from the whole business logic.

Our work aims at integrating DAOs into process-driven SOAs, so it is concerned with both processes that typically invoke services and with services that can access data. Ricken's top-down approach [12] addresses the same concern by adding service-orientation to process models to support IT developers in translating business needs into executable code. In [13] a set of architectural and software design models based on proven patterns is introduced for process-driven SOAs. Both approaches, however, do not separate different views or focus specifically on data-related views.

Akin to the approach by Wiederhold [14] our approach uses a mediator-based architecture for integrating data stored in heterogeneous database systems. As the DAO concept provides an abstract and technology-independent interface for accessing data, Wiederhold's mediators enable the homogeneous data access by integrating and selecting necessary data among different sources. In the proposal of Roth and Schwarz [15], a wrapper encapsulates the underlying data and acts as a mediator between the data source and the middleware. In contrast to these mediator approaches, we propose a more abstract, higher-level approach by using a DAO repository for managing DAO operations. Kurz et al. provide a schema-level data integration which specifies how to select, integrate, and transform data from heterogeneous data sources [16]. Like our solution, this modeling approach provides a specification in a user-friendly and platform-/language-independent way. In Section 2 we presented our architectural concept for managing DAOs in process-driven SOAs. These architectural components are supported by technology-independent view models and their technology-specific extensions. As in our approach, Marcos at al. [17] support two different aspects. They distinguish between platform independent (PIM) and platform specific models (PSM) and separate models according to distinct aspects. Besides Marcos et al. [17], there are several approaches for including software architecture into MDA platform. For example, Alti et. al [18] integrate software architecture concepts into MDA platform by a UML profile for Component-Object based Software Architecture extending UML 2.0.

Our view models extend the VbMF [6] to integrate data-related views. Mendling et al. propose a similar approach for efficient view integration. They identified formal semantic associations between elements of the process view. Just like VbMF, our data-related views, in contrast, use a name-based matching algorithm to integrate views. For the establishment of the DAO repository we were inspired from current web service registry standards such as UDDI [19], ebXML [20] and WSIL [21]. EbXML Web Service Registries [20] have interfaces that enable submission, query, and retrieval of the contents of the registry. We adopted the fundamental interface abstractions, used in these approaches, to integrate the DAO repository into our process-driven architecture.

## 6 Conclusion and Future Work

In this paper we identified current problems in managing DAO operations in process-driven SOAs. In order to efficiently manage and integrate data into process-driven SOAs, we proposed an architecture, that consists of four main components namely the BPEL Process flow, the Service Repository, Service Operation Flows, and a DAO Repository. We further provide a model-driven solution to support this architecture by specifying a set of view models. As our view models are based on VbMF, each model represents a specific view tailored to the requirements of specific stakeholders. In particular, we introduced a view model for specifying database transaction flows to extract data flows from the whole business logic. Up to now, no standard retrieval or submission interface for DAO repositories has been defined. As the number of services grows, data development complexity increases with the number of data access object operations. Hence, retrieving a particular DAO operation can be complex and time-intensive. More powerful searching capabilities, such as those that can be provided on top of our approach, are hence desirable. However further work is necessary for specifying the requirements for a DAO Repository in detail. Further work also includes runtime statistics for measuring how often a DAO operation has been invoked, etc.

**Acknowledgement.** This work was supported by the European Union FP7 project COMPAS, grant no. 215175.

## References

[1] Zdun, U., Hentrich, C., van der Aalst, W.: A survey of patterns for service-oriented architectures. International Journal of Internet Protocol Technology 1(3), 132–143 (2006)
[2] Hibernate: Hibernate (2006), http://www.hibernate.org
[3] Ibatis: Ibatis (2006-2007), http://www.ibatis.org

[4] AndroMDA: AndroMDA EJB3 Cartridge (August 2007), http://web.aanet.com.au/persabi/andromda/
[5] Fornax-Platorm: Fornax-Platform Cartridges (August 2006), http://www.fornax-platform.org/cp/display/fornax/Cartridges
[6] Tran, H., Zdun, U., Dustdar, S.: View-based and model-driven approach for reducing the development complexity in process-driven SOA. In: Abramowicz, W., Maciaszek, L.A. (eds.) Business Process and Services Computing: 1st International Conference on Business Process and Services Computing (BPSC 2007), Leipzig, Germany, September 25-26, 2007. LNI, GI, vol. 116, pp. 105–124 (2007)
[7] Völter, M., Stahl, T.: Model-Driven Software Development: Technology, Engineering, Management. Wiley, Chichester (2006)
[8] Berardi, D., Calvanese, D., Giacomo, G.D., Lenzerini, M., Mecella, M.: Automatic composition of e-services that export their behavior. In: Orlowska, M.E., Weerawarana, S., Papazoglou, M.P., Yang, J. (eds.) ICSOC 2003. LNCS, vol. 2910, pp. 43–58. Springer, Heidelberg (2003)
[9] Schkolnick, M., Tiberio, P.: Estimating the cost of updates in a relational database. ACM Trans. Database Syst. 10(2), 163–179 (1985)
[10] Eclipse: Eclipse Modeling Framework (2006), http://www.eclipse.org/emf/
[11] Apache Software Foundation: Axis2/Java (2004-2008), http://ws.apache.org/axis2/index.html
[12] Ricken, J.: Top-down modeling methodology for model-driven soa construction. In: OTM Workshops, vol. (1), pp. 323–332 (2007)
[13] Zdun, U., Dustdar, S.: Model-driven and pattern-based integration of process-driven soa models. Int. J. Business Process Integration and Management 2(2), 109–119 (2007)
[14] Wiederhold, G.: Mediators in the architecture of future information systems. Readings in agents, 185–196 (1998)
[15] Roth, M.T., Schwarz, P.M.: Don't scrap it, wrap it! a wrapper architecture for legacy data sources. In: VLDB 1997: Proceedings of the 23rd International Conference on Very Large Data Bases, pp. 266–275. Morgan Kaufmann Publishers Inc., San Francisco (1997)
[16] Kurz, S., Guppenberger, M., Freitag, B.: A uml profile for modeling schema mappings. In: ER (Workshops), pp. 53–62 (2006)
[17] Marcos, E., Acuña, C.J., Cuesta, C.E.: Integrating software architecture into a mda framework. In: Gruhn, V., Oquendo, F. (eds.) EWSA 2006. LNCS, vol. 4344, pp. 127–143. Springer, Heidelberg (2006)
[18] Alti, A., Khammaci, T., Smeda, A.: Integrating software architecture concepts into the mda platform with uml profile. Journal of Computer Science 3(10), 793–802 (2007)
[19] Clement, L., Hately, A., von Riegen, C., Rogers, T.: UDDI Version 3.0.2, UDDI Spec Technical Committee Draft (October 2004), http://www.uddi.org/pubs/uddi_v3.htm
[20] OASIS/ebXML Registry Technical Committee: OASIS/ebXML Registry Services Specification v2.0. (December 2001), http://www.ebxml.org/specs/ebrs2.pdf
[21] Ballinger, K., Brittenham, P., Malhotra, A., Nagy, W.A., Pharies, S.: Web services inspection language (ws-inspection) 1.0 (November 2001), http://www.ibm.com/developerworks/library/specification/ws-wsilspec/

# WIMS 2.0: Enabling Telecom Networks Assets in the Future Internet of Services

David Moro[1], David Lozano[1], and Manuel Macias[2]

[1] Telefónica Investigación y Desarrollo, S.A.U. Parque Tecnológico, Boecillo (Valladolid), 47151, Spain
{dmf,dll}@tid.es
[2] Full on Net, S.L. Hnos. García Noblejas, 41, 28037 Madrid, Spain
mmacias@full-on-net.com

**Abstract.** WIMS 2.0 initiative, focalized on convergence of telecom networks with web 2.0, provides mechanisms to offer the unique telecom service assets as part of the future Internet of Services. To achieve this, WIMS 2.0 technological foundation lays on the Open APIs concept: a so-called Telecom Exposure Layer at the operator network provides interfaces to easily integrate telecom capabilities with Internet services. Not withstanding, WIMS 2.0 includes other strategies: the telecom Portable Service Elements, providing a widget-based penetration of personal communication services into the Internet loom of services; the telecom-boosted User Generated Content publication and distribution, exploiting telecom services potential to produce UGC in real-time; and the Thin Clients, providing virtual terminal representations, ubiquitously accessible from any point of the Internet. WIMS 2.0 provides a mid-term view for the immediate evolution towards the future Internet of Services from a telecom-convergent view.

**Keywords:** Internet of Services, Web 2.0, telecom, convergence.

## 1 Introduction

The present paper condenses the ideas developed in the WIMS 2.0 (Web 2.0 & IMS) initiative. Taking into account the Web 2.0 revolution [2] and the future trends of the Internet of Services [1], WIMS 2.0 proposes an ecosystem for the creation of innovative user-centric services through the provision and exposure of unique telecom operator service assets towards the Internet.

Recently, a new philosophy for the design and development of services has brought about a massive wave of highly-social services in the Internet, creating the current Web 2.0. In this new fashion, the user becomes the centre of the service and the Internet becomes the platform for developing and delivering services, in many cases, through the mixture of functionalities (mashups) and content (syndication) coming from different Internet players. Continuing this approach, the future Internet will follow an open garden philosophy, with a flexible, cost-effective, technology independent deployment of new services in a plug-and-play service activation fashion.

© Springer-Verlag Berlin Heidelberg 2008

In order to adapt to this strategy, telecom operators must move from a network-centric to a user-centric approach and concentrate on the role of “Enabling Platform Providers for the Construction of Services”. The appropriate exposure of operator’s capabilities and platforms towards the Internet will enable the flexible and faster construction of final services by Internet players (e.g., a service provider on the Web or a user generating their own applications).

WIMS 2.0 has identified several strategies for exposing telecom capabilities, ideally based on the IP Multimedia Subsystem (IMS) [3] but not limited to it, from the operator networks to the Internet. Regarding IMS, it has been considered as the basis platform to support the exposure of telecom capabilities due to several reasons. Specifically, IMS is completely based on IP technologies and it was originally designed to support multimedia communications. Also, IMS-based set of standardized service enablers will conveniently enrich the capabilities to be exposed. In addition, apart from the above mentioned advantages, IMS will become the common control system for mobile, fixed and convergent telecom networks and services; thus, the consideration of IMS allows conceiving the telecom capabilities exposure strategy from a general point of view, able to address a widely common telecom operator scenario.

In the first place, in order to illustrate the context of utilisation and motivations of the WIMS 2.0 initiative, this paper shows some of the relevant characteristics of the future Internet. Afterwards, WIMS 2.0 convergence strategies are exposed and organized in different guidelines. Later, the reference architectural model, defined to obtain the desired convergence along the proposed guidelines, is presented. This reference model allows establishing the main architectural concepts for the definition of a specific WIMS 2.0 Service Platform. Finally, and aiming to clarify the viability of the proposed solution, this paper presents the results of one representative proof of concept within the WIMS 2.0 initiative. The positive outcomes and the interest of several WIMS 2.0 proofs of concept and use cases have prompted WIMS 2.0 founders to start a practical implementation of the presented ideas.

## 2 The Future of the Internet

The future Internet [1] is envisioned as an open garden of services, where new models will be developed for the deployment, distribution and discovery of services in a efficient, controlled and network technology independent way. In this context, Internet will become a flexible open environment for the deployment of new services in an easy plug & play service activation fashion.

In this context, combination and flexibility becomes major principles for the design and development of final services, being currently essentials of the Web 2.0 revolution as the first step of the future Internet of Services. In this respect, the worldwide adoption of Internet and IP connection capabilities, along with the appropriate use of remote procedure execution schemes (i.e. open Web APIs), is enabling the mixture of service functionalities (mashups) and content (syndication). The possibility to combine resources coming from different locations of the Internet, as if it were a huge computer, provides an extraordinary flexibility to create new services in a short time. Thus, the Internet becomes the platform for developing and delivering new cost-effective services, virtually to any part of the world.

In addition, a new philosophy based on user-centric approach is emerging in the design and development of new services. In this paradigm the user is the centre: the user is now regarded as the main active driver of the service, so that he can freely express his preferences. The users create the service content, they can customize service features, they effectively affect the service evolution and they can even participate in service development, directly constructing modules and applications in order to fulfill their own needs.

Additionally, the new universe of forthcoming services will effectively consider the situation and context of users, capturing their environment and localization at each time. These new services will provide extended (multimode, ubiquitous, personal, contextual, pro-active, self-learning, etc) and multi-device interaction with users, bringing forward the preferences and necessities of users at each moment. Besides, the future of internet will be associated with mobility, dynamism and advanced connectivity. Always-on connectivity will be essential, with services always available independently of the device and access technology (fixed or mobile, etc).

Taking into account some of these future trends, telecom operators must adapt their service architecture and business models for converging, being involved and provide advanced service connectivity for the future Internet. Thus, telecom operators must implement a service architecture based on the offer of network operator capabilities and provision of means for users to be always connected and interacting with the future service-oriented Internet.

## 3 Exposure of Telecom Operator Service Assets to the Internet

The main objective of WIMS 2.0 is to make the telecom network operator services available into the future Internet of Services, being the first step to make them available into the current Web 2.0. Following this approach, the final service is actually provided by a Third Party located in the Internet, but the operator offers an added value, maintaining an active role through the provision of unique telecom service assets, apart from providing connectivity. WIMS 2.0 proposes a strategy organised in several guidelines. The first three general guidelines are considered within WIMS 2.0 initiative to cover this approach:

**Guideline 1: Incorporation of IMS capabilities into the Internet of Services through open Web APIs**, allowing the integration of IMS and telco capabilities into the Internet of Services, currently enabling this integration into Web 2.0 mashups. This potentially applies to any IMS capability and by extension to any telecom service capability, which would therefore be usable by/from any Internet service. Two different, but interrelated, strategies are proposed here:

1. **Portable Service Elements (PSEs):** IMS and telco applications can be incrusted into the Internet of Services in the form of web-widgets, which would be the technology-specific implementation of a PSE. The PSEs provided by the operator (or a Third Party) can be easily integrated by end users into current Web 2.0 sites and, once incrusted, they are able to handle the interaction with the operator's open Web APIs, thus, enabling the use of IMS communication capabilities from Web 2.0 sites.

2. **Direct incorporation into Internet mashups:** Once IMS and telco capabilities are exposed through open Web APIs, they can be directly incorporated into any service of the Internet in order to provide complementary functionalities, just like any other regular mashup. Of course, the inclusion of IMS and other telco capabilities is a decision of the Internet service (i.e. the Third Party); however, the final outcome is a complete integration of telecom capabilities in the resulting service. Due to the fact that current Web 2.0 service operation must be modified to consider the use of IMS APIs, the application of this strategy is not immediate, but for the mid and long-term it enables a convergence of great impact and further applicability.

**Guideline 2: Publication of user-generated content enabled by IMS.** Mobile handsets are expected to be one of the main sources of multimedia content in the future. This convergence guideline aims to obtain new ways for content publication through the usage of IMS or legacy multimedia transmission capabilities. Operators can provide new solutions to receive the multimedia content from the user and automatically upload it to Internet and currently more concretely to the Web 2.0 sites. An example application is "YouTube real-time video generation", where users video-call YouTube to create a spontaneous clip. Other examples may explore the use of other types of user-generated data, such as IMS Presence or other user's context information. By extension, pre-IMS capabilities like e.g. videotelephony are good candidates to be considered in this guideline too.

**Guideline 3: IMS on-line applications.** Through the use of AJAX and other technologies, current Web 2.0 applications have achieved major advances in the field of user interfaces. WIMS 2.0 favours for the usage of these technologies in order to build on-line application that directly use the operator's communication services. The objective of this convergence guideline is to achieve that web applications act as the user terminal. The benefits of such kind of IMS/telco applications are, mainly, ubiquity and a great simplification of service development and deployment, since the client service logic actually resides on the operator's network. An example of application is to enable a virtual IMS terminal within an end-user's browser.

Additionally WIMS 2.0 incorporate a fourth guideline for the integration of generic multimedia content and events from the Internet within telco services. This strategy will enhance end-user experience, completing in this way Internet and telecom network operator convergence:

**Guideline 4: Incorporation of future Internet content and events into telecom services.** Since Web 2.0 services are already holding the successful content (the user-generated content) the WIMS 2.0 initiative champions new mechanisms and functionalities for obtaining generic multimedia content and events from the Internet. Among the content, we may find videos, advertisements, podcasts, news, etc. Among the events, we may find contextual information associated to social networks, new content publication alerts, etc. The introduction, integration and distribution of all this information within IMS services is of great interest, since it can enhance end-user's satisfaction. As example services, we may consider special feed readers for mobile handset, the inclusion of social networks events into IMS Presence or Web 2.0 video and music automatic distribution using IMS/legacy multimedia transmission capabilities.

## 4 A Reference Model for the WIMS 2.0 Service Platform

This section introduces a reference model for the WIMS 2.0 service architecture. The proposed reference model covers the WIMS 2.0 strategic convergence guidelines presented above. From a high level point of view, the WIMS 2.0 service architecture must be a service platform that, acting as an intermediary for the adaptation between the operator network and the Internet, enables the convergence in the desired terms. The proposed reference model is reflected below in Fig. 1.

The above reference model defines a framework that identifies the required logical entities, as well as the relationships existing among them. Due to the different natures of the systems to be converged, this level of abstraction permits to settle the main concepts and design principles prior to the definition of a concrete service architecture considering specific technologies.

Two main groups of entities can be identified in the WIMS 2.0 reference model: entities exposing telco capabilities to the Internet of Services, including the ones providing IMS/telco-based online applications to the end-user's browser, and entities for the exchange of multimedia content and events between the Internet of Services and the operator network. For these two groups, a common entity is set as the axis of the reference model, the IMS Exposure Layer entity for exposing telco and IMS capabilities. For clarity reasons, we will present this basic entity within the first group.

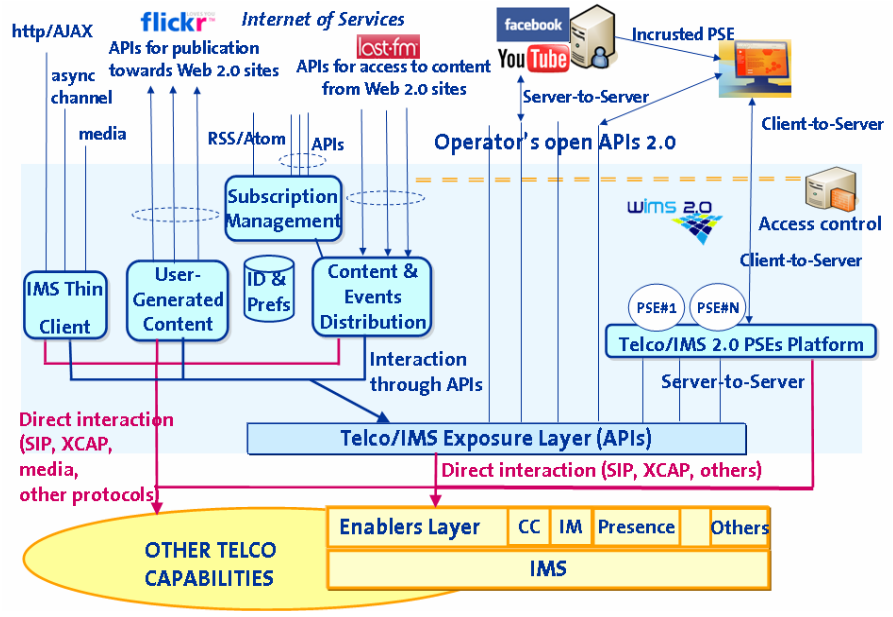


**Fig. 1.** Architectural reference model for the WIMS 2.0 service platform

The entities dedicated to the exposure of telco capabilities to the Internet, and their functionalities are presented below:

- **IMS Exposure Layer:** The entity for the exposure of IMS capabilities is the IMS Exposure Layer. This entity exposes IMS capabilities to the outer world through open Web APIs. Concretely, WIMS 2.0 focuses on the use of REST (Representational State Transfer) [4][5] for these APIs, since it is simpler and lighter than other RPC-based approaches (e.g. SOAP Web Services) [6] and, more importantly, REST APIs are the predominant technology in today's Internet mashups. Therefore, on the outer side, the IMS Exposure Layer manages RESTful HTTP messages [11] requesting or responding to the execution of a specific procedure [7], while, on the inner side, it interacts with IMS/telco capabilities using the appropriate protocol (SIP [8] or XCAP [9]). As it can be noticed, it is actually a gateway between HTTP and other telco protocols. In principle, all the available IMS/telco capabilities can and should be exposed by this entity following this approach.
- **IMS 2.0 PSEs Platform:** This entity hosts and serves telco widgets or PSEs to be incrusted into Internet sites (user's browsers). These PSEs are provided by the operator and they offer telecom service features using the operator's APIs exposed to the Internet of Services environment. This platform can expose simplified APIs and translate the interactions through those APIs towards the "complete" APIs exposed by the IMS Exposure Layer. IMS 2.0 PSE may also access the telco capabilities directly.
- **IMS/Telco Thin client:** Entity for the delivery of IMS on-line applications. On one side, this entity interacts with IMS/Telco capabilities, acting as the final IMS/Telco client. On the other side, it shows a rich multimedia Web interface to the user, according to the service being provided. This entity must also handle a continuous media interface towards the user, as well as an asynchronous interface to signal the need to refresh the Web interface due to an incoming network event. As the final result, this entity enables the virtual terminal within the end-user's browser.
- **Access control entity:** To secure the use of the operator's APIs, performing identity authorization and authentication and parameters conversions.

The entities for the exchange of multimedia content and events, and their functionalities, are the following:

- **User-Generated Content enabler:** This is the most relevant entity in terms of providing user generated content in real time towards internet services. This entity receives content from IMS/Telco terminals and, after adapting its format, it uploads this content to the Internet, and more concretely currently to Web 2.0 services. To receive the content from the user, it uses a specific set of IMS capabilities.
- **Subscription Management enabler:** It subscribes to content and events of any kind generated in the Internet of Services. Currently it subscribes through RSS/Atom [10] channels to contents and events of Web 2.0 services. Subscriptions may be made on behalf of the operator or directly on behalf of the final user. This entity talks with the Content & Events Distribution enabler to inform about the existence of new content/events to be obtained.

**- Content & Events Distribution enabler:** It downloads, adapts and transmits content and events from the Internet of Services to IMS/Telco users. To perform the transmission on the telecom network side, it uses a specific set of IMS/Telco capabilities.
**- IDs & Preferences:** Database that stores the relationships between IMS identities and the identities used by the Internet of Services. The other entities of this group consult this database to perform the conversion of identities when using Internet services APIs.

In this model, the IMS Exposure Layer is of paramount importance since it provides a basis for the rest of entities to access the telco capabilities in a simple way, while permitting other external applications in the Internet of Services ecosystem to access the telco environment and its service creation assets. Thus, the IMS Exposure Layer is responsible for enabling the so-called network mashups combining functionalities from the Internet and the telco world. The use of RESTful APIs in the IMS Exposure Layer provides non-telco developers with an understandable and familiar view of the telco service capabilities, which is ultimately intended for the promotion of its extensive use within the Web 2.0 world.

The proposed reference model has been tested through several proofs of concept: IMS communication widgets, telco-enabled mashups based on IMS APIs, and content publication and distribution services that use telco transmission capabilities for exchanging Web 2.0 contents. The result of these proofs of concept have validated the correctness of the proposed architecture, although a high capacity, industrial environment still needs to be developed in order to test the scalability of the system. An example of such proofs of concept is presented in the next section.

## 5 Proof of Concept: Mashing up Telecom Services with Web-Oriented Applications. The FindUs Application for Social Networks

In order to illustrate the WIMS 2.0 concept, a use case "Find Us Application for social networks" has been implemented. This use case is in the scope of the first guideline of the WIMS 2.0 service strategy presented above: Incorporation of IMS capabilities into the Internet of Services through open Web APIs, featured for the current Web 2.0. Specifically, this use case is materialized as a gadget application which implements a mashup re-mixing several external functionalities/services. The gadget application is included in a social network container. In this example, the host social network is the popular Facebook. In this social network environment, this kind of gadget applications are not provided by the social platform operator, but rather by end-users. Even though the end-users are required to entitle some developing skills, this ratifies the user-centric approach as one of the axis of the new-era Internet. In the near future, developing tools for User Generated Services will allow almost anyone to synthesize smart applications by combining available service resources in the Internet of Services.

The end-user service concept of the FindUs application is shown in Fig. 2 below. As shown, the gadget is embedded in Facebook, thus user interface of the application

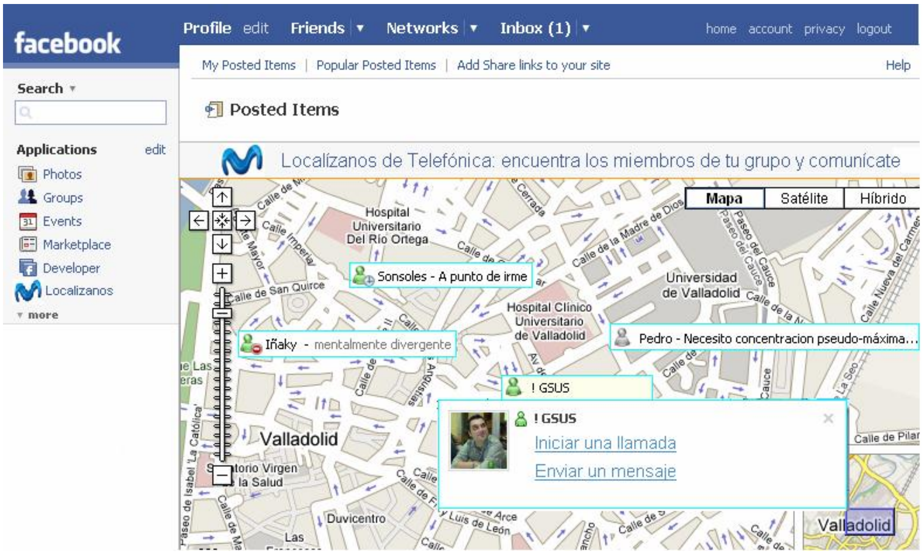

facebook
Profile edit Friends Networks Inbox (1)
home account privacy logout
My Posted Items | Popular Posted Items | Add Share links to your site
Help
Posted Items
Search
Applications edit
Photos
Groups
Events
Marketplace
Developer
Localizanos
more
Localízanos de Telefónica: encuentra los miembros de tu grupo y comunícate
Mapa Satélite Híbrido
Sonsoles - A punto de irme
Iñaky - mentalmente divergente
Pedro - Necesito concentracion pseudo-máxima...
! GSUS
Iniciar una llamada
Enviar un mensaje
Valladolid

**Fig. 2.** Service concept of FindUs gadget application

is rendered by web technologies. With this application, a social network user can locate his/her group of friends in a map and if available, their position is calculated utilizing mobile location. Alternatively, the civil address included in the social network user profile is utilized and marked as such. For each located user, an icon is drawn representing the user's current presence communicating status and the personal note describing current user context. This personal note is normally imputed by the user in his/her mobile terminal or Communicator application and automatically obtained by the FindUs application.

When clicking on any of his/her located contacts, the user can obtain a list of service features that can be invoked to communicate with the located users. For instance, the user can request a click-to-call service to establish a communication between his/her mobile phone and the located user's mobile phone. Alternatively, the user might be able to start an instant messaging conversation from the web interface of the gadget towards the located user's mobile phone, or just send a single message to this located user.

In the specific case of the FindUs gadget application, several resources from a current instantiation of the Future Internet of Services are utilized. The application is mashing up programmatically-controllable Maps from Google and User Profile data provided by Facebook with telecom service assets like IMS Presence, IMS Third Party Call, IMS IP messaging and cellular location. All these service resources are made available to the wide Internet by means of Open APIs interfaces, which expose the functionality and perform service invocation. Figure 3 shows this new paradigm for service creation with the utilization of telco operator service assets as service resources available in the Internet. Concretely, in our example, telco services assets consisting of person-to-person conversational capabilities (i.e. click to call), user context information (i.e. presence) and messaging capabilities (i.e. instant messaging)

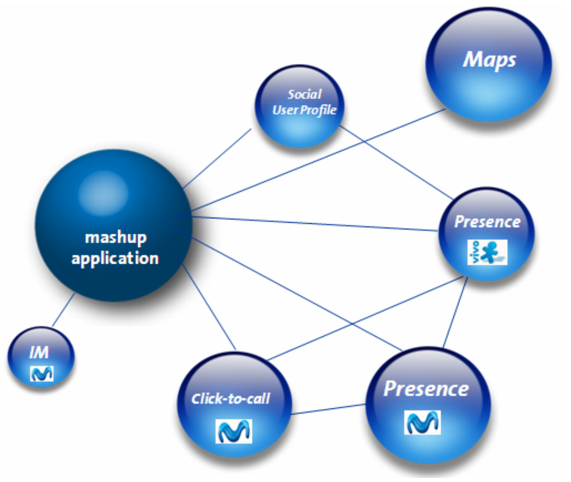


**Fig. 3.** Example of the Internet of Services for the FindUs gadget

are exposed and made available for combination together with other service resources across the Internet, such as the social user profile information from Facebook, maps from Google or Presence information from an Internet Server. The combination of this range of services brings about innovative and enhanced mashup applications also able to be combined and used as service resources.

Regarding the proof of concept implementation, Fig. 4 shows a high level overview of the interworking between the entities from the Web 2.0 and the operator network that supports the present use case. From the operator network side, some of the entities from the WIMS 2.0 reference model presented above are involved. Concretely: the Telco/IMS Exposure Layer, the Telco/IMS PSEs Platform and the Access Control together with a series of enablers (i.e. location, presence, telephony and instant messaging) that finally provide the telco capabilities. In order to avoid current State of the Art restrictions with regard to telco capabilities, enablers are not mandatory based on IMS, being pre-IMS telco enablers also considered in this implementation. This fact validates and remarks that WIMS 2.0 is viable for general Telco environments, not just for those based on IMS.

For this use case, the FindUs application is hosted within the PSEs platform, which is in charge of invoking the necessary telco APIs (usually based on REST technology) and other Web APIs from the Internet. In this scenario, when a Facebook user starts a session in Facebook and executes the FindUs application, the Facebook platform acts as a merely HTML traffic intermediary, forwarding requests and responses between the user's browser and the PSEs platform. As the first step in order to provide the FindUs functionality previously presented, the PSEs platform invokes a particular Facebook Rest API including the Facebook credentials of the user to obtain the contact details of his/her group of friends. Subsequently, when the telephone number from each friend of the group is retrieved, the PSEs platform invokes the cellular location API exposed by the Telco/IMS Exposure Layer. This location information will be forwarded back to the user's browser, being .the location of any friend of the

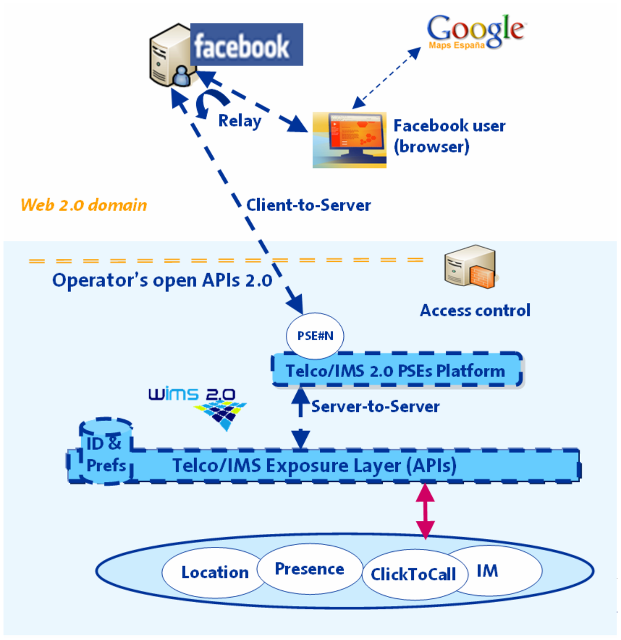


**Fig. 4.** High level overview of the interworking between Web 2.0 and WIMS 2.0 Reference Model for "The FindUs" application

group available at the user's browser. Finally the user's browser will access the REST API from Google Maps, to obtain the maps for the graphical representation of the location of each friend of the group. A similar procedure will apply for the case of retrieving friends' presence information from the operator network, where PSEs platform will invoke the presence API of the Telco/IMS Exposure Layer. Regarding, the remaining telco functionalities, for instance, if the user wants to starts a conversational or instant messaging session, the PSE platform will send the necessary requests towards the appropriate telco capabilities exposed by the Telco/IMS Exposure layer.

This proof of concept clearly reflects the power of a typical Web 2.0 concept that, this time, is applied in the telco environment: service mashups. The combined used of different Open Web APIs has enabled the fast creation of a service that mixed standard-web services, like social User Profile and Maps, with IMS/Telco communication and user context features. As it can be observed, this represents a powerful tool for creating new application by combining resources made available in the Internet of Services.

## 6 Conclusions

The future internet is envisioned as an open garden of services, where services will be deployed easily in a plug & play service activation fashion, resulting from the flexible combination of resources and services from anywhere. In this context, telecom operators must follow this new philosophy adapting their infrastructure and business model with new mechanism for exposing their unique service assets as service resources towards the Internet. Thus, operators will integrate into the future of internet value chain in order to be not only a connectivity provider but also an added-value provider in the creation of innovative services.

WIMS 2.0 establishes a series of strategies for efficiently exposing telecom operator network services to the future Internet of Services and for enabling the convergence of telco and Internet worlds. WIMS 2.0 is based on the aperture of telecom capabilities through Open APIs providing web friendly interfaces to easily integrate telecom capabilities with Internet services. Together with this Open API strategy, WIMS 2.0 establishes additional strategies for convergence that include: Portable Service Elements, providing a widget-type penetration of personal communication into Internet applications, User Generated Content publication and distribution for exporting UGC in real time, and the so-.called Thin Clients, providing telco-based online applications, like virtual terminal representations, ubiquitously accessible from any point of the Internet.

To materialize this convergence approach, a reference model for the WIMS 2.0 Services Platform has been proposed following the general idea of providing an intermediate layer to break the frontiers and adapt the interactions between telco and Internet domains. This core reference model mainly lies upon the exposure of telco/IMS capabilities via open APIs, based on a Web 2.0-friendly approach in order to maximize the effective and widespread adoption. Besides the entities exposing telco capabilities to the Internet, this reference model includes several entities for the exchange of multimedia content and events between the Internet and the operator network to establish a full convergence.

Finally, one proof of concept, based on the strategy of incorporating and remixing IMS capabilities along with other Web 2.0 services through open Web APIs has been implemented and presented in this paper, as an example to illustrate and validate the potential of the WIMS 2.0 principles.

## References

1. European Commission Information Society and Media: The future of the Internet. Area 2, Service Architectures, Brussels, Belgium, pp. 54–55 (2008)
2. O'Reilly, T.: What is Web 2.0? Design Patterns and Business Models for the Next Generation of Software, http://www.oreilly.com/pub/a/oreilly/tim/news/2005/09/30/what-is-Web-20.html
3. 3GPP TS 23.228 v 8.3.0, IMS: IP Multimedia Subsystem, Stage 2, http://www.3gpp.org
4. Fielding, R.T.: Architectural Styles and the Design of Network-based Software Architectures. Ph.D Thesis, ch.5. University of California, Irvine (2000)
5. Fielding, R.T., Taylor, R.N.: Principled Design of the Modern Web Architecture. In: Proceedings of the 22nd international conference on Software engineering, Limerik, Ireland, pp. 407–416 (June 2000)
6. Pautasso, C., Zimmermann, O., Leymann, F.: RESTful Web Services vs. "Big" Web Services: Making the Right Architectural Decision. In: WWW 2008, Beijing, China, April 21–25, 2008. ACM, New York (2008)

7. Lozano, D., Galindo, L.A., García, L.: WIMS 2.0: Converging IMS and Web 2.0. Designing REST APIs for the exposure of session-based IMS capabilities. In: International Conference and Exhibition on Next Generation Mobile Applications, Services and Technologies, Cardiff, Wales, UK, September 6-19 (2008)
8. Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., Schooler, E.: SIP: Session Initiation Protocol, IETF RFC 3261 (June 2002), http://www.ietf.org/rfc/rfc3261.txt
9. Rosenberg, J.: The Extensible Markup Language (XML) Configuration Access Protocol (XCAP) (May 2007), http://www.ietf.org/rfc/rfc4825.txt
10. Nottingham, M., Sayre, R.: The Atom Syndication Format. IETF RFC 4287 (December 2005), http://www.ietf.org/rfc/rfc4287.txt
11. Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T.: Hypertext Transfer Protocol –HTTP 1.1. IETF RFC 2616 (June 2006), http://www.ietf.org/rfc/rfc2616.txt

# Describing Next Generation Communication Services: A Usage Perspective

Emmanuel Bertin[1] and Noel Crespi[2]

[1] Orange Labs - France Télécom R&D - 42 rue des Coutures 14066 Caen, France
emmanuel.bertin@orange-ftgroup.com
[2] TELECOM SudParis - 9 Rue Charles Fourier, 91011, Evry Cedex, France
noel.crespi@it-sudparis.eu

**Abstract.** Telecom services are usually described from an applicative perspective. Service providers should yet describe formally what their services do for their users, in order to adapt them to user's needs and to compose them. We propose here a framework to describe communication services from a usage perspective, by describing formally the actions of the service users and of the service provider. This description is then used as a common library to compose new services and to check the consistence of these composed services.

## 1 Introduction

Telecom services evolve toward Next Generation Communication Services, also named user centric services, designed to fulfil user's needs instead of focusing on technologies and protocols. Similar evolutions are ongoing in multiple service area, like healthcare or education services. As user's needs are various and diverse, a single service can not be suitable for all communication needs: Different services must cooperate. How can this cooperation be achieved?

First, as learnt from software engineering, the various services should be kept separated with explicit boundaries, and not tightly integrated. This is usually referred as the loose coupling principle. Each service is responsible for a give task, and offers a public and well defined interface to other services. In addition, cooperation between services requires a common understanding of what each service do, i.e. a shared service description. Current service description languages like WSDL (Web Services Description Language) are focused on software aspects (API description) and not on what achieve the service for its user. This lack is especially underlined with communication services, where the value is not in data treatment, but in the quality of the user interaction and in the reliability of the executed tasks.

In this paper, we propose to describe communication services from the user point of view, by describing the actions from both the service users and the service provider. This paper is structured as follows. In section 2, we briefly survey the evolution toward user-centric services. We then study in section 3 the related work on the description of communication services. Next, we introduce in section 4 the notion of action to describe services and illustrate it step by step with the email example. The last section provides some conclusive remarks.

© Springer-Verlag Berlin Heidelberg 2008

## 2 Toward User Centric Services

### 2.1 Trends and Motivations

Communication services are part of a broader story; their usage evolves in accordance with the transformations of the whole service sector. In addition to information and telecommunication, the service sector at large includes for example healthcare and social assistance, educational services, finance and insurance or public administration, as classified for instance in the North American Industry Classification System [1]. The current and future economic growth of the developed nations is mainly driven by the service sector, as reminded in Table 1.

**Table 1.** Sectoral shares in employment, Developed Economies and European Union [2]

| | Employment in sector as share of total employment | |
|---|---|---|
| Year | 1997 | 2007 |
| Agriculture sector | 6.1% | 3.9% |
| Industry sector | 28.3% | 24.5% |
| Services sector | 65.6% | 71.5% |

These rates witness our societal transformation from an industry society into a service society, as surveyed for example in [3], [4] or [5]. In an industry society, driven by mass market production, even services are considered as products, developed and marketed like products. In our service society, services are driven by individual and specific needs. As a single service entity is not able to meet alone the needs of users, a service is more and more considered as a composition of basic services. We observe this evolution in various social fields like healthcare services, education services or employment services: the trend everywhere is to enhance the "user experience" by making cooperating different entities to meet a unique user need [6]. The same trend can be noticed in IT.

According to this trend, service organizations are shifting from a product-centric paradigm to a customer/user-centric paradigm, as surveyed for example in 2006 by Shah [7], Day [8] or Kellogg [9]. Where the product-centric approach consists basically in selling products to whomever is ready to buy it, the customer/user-centric approach consists in serving customers/users by taking into account their needs. Rust [10] underlines even that "we witness the rise of “mass service,” driven by improved coordination and a greater availability of information. Whereas mass production focused on the product, the new philosophy is customer-centric."

### 2.2 User Versus Customer

We should draw here a distinction between users and customers. When considered with a product-oriented mindset, a service is seen as a delivery according to a client order. When considered with a service-oriented mindset, a service is seen as a value co-creation process between the service consumer and the service provider, as established clearly by Tien [11] or Spohrer [12]. The more recent researches on services emphasize indeed that the user is directly implied in the value creation process of the service.

This is even more accurate with the rise of the audience economical models, where services are free and where service providers are remunerated by advertisers. The value of a service for a service provider is thus correlated with its capacity to drain advertising incomes. This capacity is itself correlated with the value of the service for users. In an audience model, the primary actor is thus no more the customer (the one who pays the bill), but the user (the one who sees the ads). Service providers should then consider the usefulness of their services for users. We have thus to investigate what is a service as seen by the user, and not only as designed or operated by the provider.

### 2.3 Services as Systems

Services at large are studied in marketing, in sociology and in organization management studies. Some of these studies are now merging with IT studies around a thematic named service science or SSME for Services Sciences, Management and Engineering [13].

Services are not standalone and tangible entities like products. A product does exist without clients, but a service does not exist without users. This is usually referred as the inseparability of production and consumption [14]: A service can not be considered independently of its consumption, of its usage. To reflect this, many authors envisage a service as a system [11, 12] composed from business actors (user and provider), from products and technologies (including hardware, software, protocols), and from a service logic (or service processes) linking the whole. This system is connected with other service systems.

Now, let us come back to communication services. First, communication services are evolving as presented above: they are more and more considered as a composition of basic services, they more and more follow a user-centricity paradigm, they are integrated in a whole service system. In addition, communication services are interactive services. Following [15], communication services can thus be classified as user-intensive. This means that the user provides significant inputs in the service logic.

We focus in this paper on the service logic that links the service user and the service provider, taking into account the interactive nature of communication services. We propose a high-level service description method to specify the interactions of users and providers. Let us review now the existing ways to describe communication services.

## 3 Existing Approaches to Describe Telecom Services

Describing telecom services is not a novel issue. As surveyed recently in [16], this has been a recurrent task in the telecom world, first inside the Intelligent Network paradigm, then through the TINA-C (Telecommunications Information Networking

Architecture consortium) initiative at the end of the 90ies, and more recently at ITU-T, ETSI or at the OMA (Open Mobile Alliance) [17]. At the same time, in the IT world, the service description issue has been mainly considered through the SOA (Service Oriented Architecture) paradigm, and the SOA approach is now percolating through to telecom services. Moreover, the semantic web community has also focused on this issue and its methods are now considered for describing telecom services, for example in European projects like Spice [18].

### 3.1 Telco Initiatives

Inside the telecom world, the first comprehensive initiative for modeling services has been the Intelligent Network, developed in the 80ies. A service is viewed from a user perspective as a collection of service features that complement a teleservice, where "a teleservice is a type of service that provides the complete capability, including terminal equipment functions, for communication between users" [19]. For example, the Call Forwarding feature complements the telephony service. Theoretically, this model enables to compose service feature to form new services, but, as acknowledged in [16], this possibility was never exploited. Norms mention service features nearly anecdotally, without defining structuring rules or composition rules. Service feature are defined as significant functions from the user point of view, but why and how these functions are significant is not clear. In summary, telecom operators and vendors have forged and have used the IN concepts with a product-oriented mindset, to sell nearly standardized services where users were interchangeable.

The TINA initiative tried to overtake the IN shortcomings, but did not specify a high-level service description language, focusing rather on a generic service session concept that should fit to every kind of services.

As surveyed in [20], ITU-T, ETSI and OMA have introduced more recently the concept of service building block. These service building blocks are called "Service Capabilities" by the 3GPP, "Service Support Capabilities" by the ITU-T and "Service Enablers" by the OMA. Service Support Capabilities studied at the ITU-T [21] typically include presence, location, group management, message handling, broadcast/multicast, push and session handling or device management. Service Enablers at the OMA [22] include for example data synchronization, device management, digital rights management, downloading, e-mail notification, instant messaging, presence and mobile location or multimedia messaging. Service capabilities defined at the 3GPP typically include presence [23] and messaging [24] or conferencing [25]. The functional implementation of these service building blocks is described in the according standards. But there is no high-level description method to specify the added value for the end-user and to position these building blocks one over the other.

### 3.2 IT Initiatives

Inside the IT world, innovation is mostly driven by the Information Systems (IS) evolutions. In order to adapt their IS to the service era, companies had to break the boundaries between their various applications [26]. The IT world has forged the Service Oriented Architecture (SOA) paradigm to overcome the lack of cooperation between various software applications. Applications should no more be considered as standalone entities, but divided into services, i.e. discrete and independent units of business that perform a specific task and that are only accessible through an open and well-defined service interface.

Companies have discovered that the main challenge to apply this SOA paradigm was not a technical challenge. The main issue is indeed to identify and define the services, these discrete and independent units of business. Which part of the business should be considered as services? Should the services be fine grained (one function per service) or coarse grained (many functions per services)? How to ensure independence between services? How to ensure that the services suit to the enterprise business and strategy? How to identify the services that are necessary to meet a specific need? In the enterprise IT context, these questions may be answered by considering the enterprise business processes, as they describe the internal activity of the enterprise. Nevertheless, the services offered to end-users are usually not described through business processes, as indicated in [27]. With a product-oriented mindset, the main assets of a firm are indeed the efficiency of its internal processes, and not the service it offers to its users.

Moreover, when building and composing such IT services, most software engineers tend, in the end, to consider the user as a system component, providing inputs and requesting outputs like a software component (as surveyed for instance in [28]). This is not really a trouble for data services, which goal is to provide data that are treated to fulfill user needs. But concerning communication services, the complexity and the value of the service do not rely in the data treatment, but in the exchanges between the service users (e.g. caller and callee) though the service provider and in the way these exchanges are presented to the user, as detailed in [29]. As a consequence, communication services engineers are usually attaching great importance to protocols (like SIP, SMTP…) that describes the exchanges between parties.

### 3.3 Semantic Web Initiatives

The semantic web community has also widely studied mechanisms for service description, taking into account the added value of the service, as surveyed for instance by Zhixiong [30] or Arroyo [31]. Projects like Spice [18] aim to build a marketplace of services, where a user can request a service in (nearly) natural language. A service is then constructed according to his needs, by composing the existing services of the marketplace. The value is here more brought by the marketplace and its users, than by the service providers that become indeed interchangeable.

We saw above that the role of the user is shrunk when a service is considered with a product-oriented mindset. With this semantic marketplace, the role of the service provider is shrunk instead, the key actor being the marketplace provider that supplies a way to access to service resources. If this model seems effective for data services (e.g. with search engines like Google), it is not the case for communication services. Gmail is for example a "classical" email service provider and not something like an "emailing marketplace". Community and user experience are essential for communication services. As mentioned before, the value of communication services do not come from their data assets, but from the exchanges between three parties: the service user, the service provider and other service users that communicates with the first one. This three-party model cannot be decomposed into 2 two-party models. In other words, the network effect is important for communication services, as illustrated for example recently with Skype or Facebook.

In addition, like in the IN approach of service features, there is no method to identify and to classify what do a service. One could argue that this classification can be done automatically through ontology mechanisms, as the features of a service are described using a semantic language. But as surveyed by Bedini in [32], such automated tools are indeed really pertinent when they build up on an existing classification.

Finally, the integration upon the existing services and platforms is definitively a tough issue, both technically and functionally. Technically, the introduction of new technologies and software tools is required (e.g. for ontology). Functionally, the web semantic paradigm follows an open world assumption, as described for instance in [33] that is not easily compatible with a component based architecture, where component are loosely-coupled (each component is a "closed world", a black box that offers services to other components).

## 4 Modeling Service Actions

Inside the SOA or semantic web paradigms, some studies attempt to link the offered services and the needs of the users, in order to achieve an automatic matching between users and services. However, these studies largely fail to model user needs because their variety and diversity. No framework can model in detail the needs of a human being and high level hierarchy like the Maslow pyramid (as proposed for instance in [34]) are not very useful to match precisely needs and services.

To take into account the previously mentioned shortcomings, we propose to describe a service through the exchange between the service user and the service provider. By representing formally such exchanges, we intend to describe in a formal way the added value of the service for its end-users (as mentioned, a service may imply several users, for example caller and callee for a telephony service). Our service description is based on the human language, which is a shared institution, rather than on human needs or goals.

### 4.1 User and Provider Actions

In our view, the concept of action is the most adapted tool to describe this exchange between service user and service provider. Service user and service providers interact by performing actions. For instance, a caller requests a phone call and the telephony provider then delivers the call to the callee. Both the call request (action of the user) and the call delivery (action of the provider) are seen by users as a part of the service.
By action of the user, we do not mean the concrete action done on the GUI (Graphical User Interface), but the immaterial activity, that the user is trying to perform. For example, the action of sending an email is not a matter of clicking on a send button (GUI), but of effectively sending an email. This accomplishment might be enabled by clicking on a button, but this GUI aspect is not considered here.

In summary, actions done by users and service providers within the course of a service are not a matter of GUIs. It is neither a question of service platforms or service infrastructure. So we have now shifted from the "how to describe what services do" question to the "how to describe the actions of users and provider within a service" question.

## 4.2 Describing Actions

We propose to establish these action descriptions on the language usage. As stated by John Searle (widely noted for his account of social reality) in [35]: "My dog has very good vision, indeed much better than mine. But I can still see things he cannot see. We can both see, for example, a man crossing a line carrying a ball. But I can see the man score a touchdown and the dog cannot (…). To see a touchdown scored he would have to be able to represent what is happening as the scoring of a touchdown, and without language he cannot do that." In our case, we can observe that actions are usually described with the same words within a given service, whatever the service provider. For example, the words "signing in" indicate the action of logging in into the provider's system, or the term "send" in a webmail context indicates the action of sending an email. We could observe the same in other European languages than English. This leads us to identify the actions that are common for communication services.

The description of the actions to consider for a given communication service are chosen according to the following criteria.

- First, their description should be a usual answer to the "what are you doing?" question or to the "what is the service provider doing" question. What are you doing? I'm writing an email. I'm checking my mailbox. I'm talking on the phone…
- Then, these actions should be known by the user as mandatory to perform the service. Delivering an email is for instance mandatory to the email service, but adding a smiley is not.

## 4.3 The Example of Email

Let us illustrate our approach with some services around email. Our first step consists in identifying actions and actors. The email service involves 3 parties: the email sender, the email service provider and the email receiver. All these 3 roles interact to perform the email service. We can identify the following actions:

- Contact selecting action (by email sender)
- Message composing action (by email sender)
- Message sending action (by email sender)
- Message delivering action (by email service provider)
- Mailbox checking action (by email receiver)
- Message reading (by email receiver)

Moreover, we observe that some actions require another action to be achieved before. For example, Message composing is required before Message sending. We describe this fact with dependencies between actions. We can establish the following dependencies for any email service:

- The message composing action requires the selection of a contact to whom the email will be sent.
- The message sending action requires the composition of a message.
- The message reading action requires both the checking of the mailbox by the receiver and the delivery of the message by the email provider.
- The mailbox checking action requires the signing in of the receiver as a principal in the email provider system (the term principal is used here according to the Liberty Alliance vocabulary (http://www.projectliberty.org) and mean someone whose identity can be authenticated).
- The signing in action requires the subscription of an email account by a customer of the email provider (signing up action).
- The message delivering action requires the validity of the email address from the receiver, and so the subscription of an email account in the email provider system.

These actions and their dependencies are then manipulated with the semi-formal UML syntax, as represented on the figure 1. Actions are modeled as UML classes. These classes are tagged with the stereotype <<XXX>>, where XXX stands for the actor that performs the action (for example <<sender>> for an action performed by the party that sends an email). The dependencies are modeled with standard UML dependencies, graphically represented with a dotted arrow. We are using the

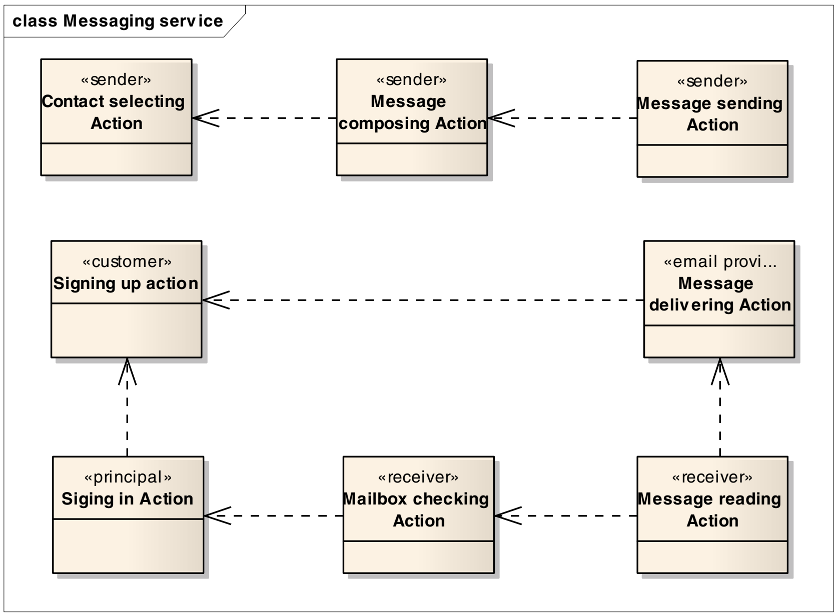

class Messaging service
«sender» Contact selecting Action
«sender» Message composing Action
«sender» Message sending Action
«customer» Signing up action
«email provi... Message delivering Action
«principal» Siging in Action
«receiver» Mailbox checking Action
«receiver» Message reading Action

**Fig. 1.** Emailing actions

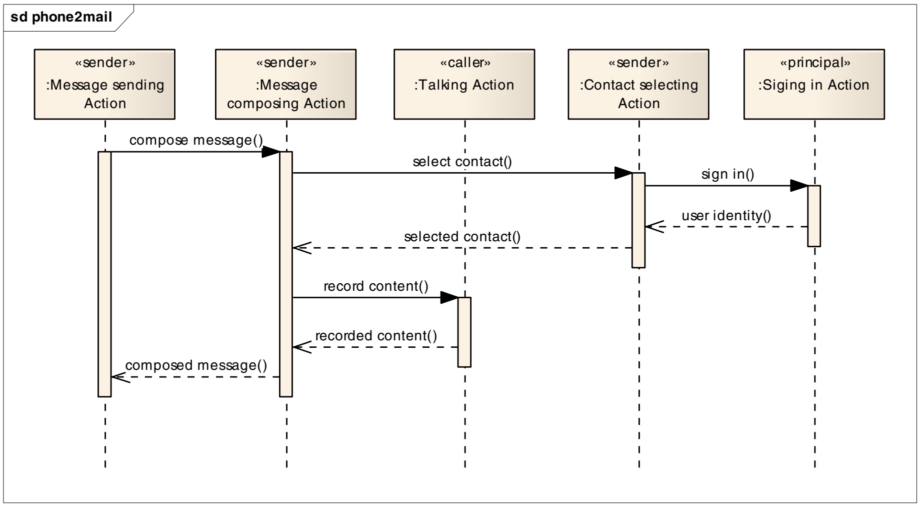


**Fig. 2.** Phone2mail actions

Enterprise Architect (from Sparx Systems) UML tool as a repository for our service descriptions. We have achieved these descriptions for the main communication services, like making and receiving calls, connecting to a communication network, sharing a personal context, signing in, signing up, watching video or hearing music, setting privacy parameters. We have identified so about forty different actions involved in the commonly used communication services.

Our second step consists in using these action descriptions as a library to describe new services. For instance, let us imagine a service to dictate emails. The service user selects a contact in his address book, triggers the service and then dictates its message, which is sent as an email to the aforementioned contact, through an attached audio file. Let us suppose that this service is marketed as the "phone2mail" service. The previous service description is rather clear but informal. It can not be shared within a formal service repository, nor interpreted by machines. With our communication action library, we can model it as an UML sequence diagram, as shown on figure 2.

The order of the sequence is not a temporal order, but a requirement order. The object at the left side of the sequence indicates the final purpose of the phone2mail service that is to send a message. The continuous arrows indicate a requirement and the dotted arrows indicate the information that is returned to fulfill the requirement. The temporal order is usually the opposite of the requirement order (the user first signs in, then selects a contact, then composes his message by talking and finally sends it).

With this formal description of service actions, we can verify the logical consistence of a service by checking if its particular chain of actions respects the general dependencies established above and represented on the figure 1 for email services. In the case of the phone2mail service, the sequence is coherent with these dependencies. For example, the phone2mail service sequence is coherent with the dependency from the Message composing action toward the Contact selecting action.

Moreover, we can also deduce from this diagram the actors of our phone2mail service. The phone2mail user should also act as a principal for authentication, as an email sender and as a phone caller. In summary, we have here represented formally the phone2mail service with a UML sequence diagram that makes use of a common action library.

Within our UML repository, we have modeled several services provided by Orange, using this common action library. This enables us, in a third step, to compare objectively these services because they use the same description method and semantic. We are now working with the marketing business units in order to make use of this repository at the business level. This will enable marketers to describe formally their offers and to compare them with existing ones. This will also enable them to browse existing offers, especially in order to reuse existing services to build new ones.

## 5 Conclusion

User centricity is a key challenge for services in general and for next-generation communication services in particular. In order to adapt services to user's needs and to compose them, service providers should be able to describe formally what their services do for their users. We propose here to achieve this goal by describing formally the actions of service users and of service providers, using a common library of actions. This way to describe services is worked with marketers in order to build a conceptual tool that suit to their needs. They can in particular build new services by recomposing existing actions and check the consistence of this service according to existing logical dependencies between actions.

We plan to further investigate two topics. First, we are going to link these service descriptions with technologies like IMS (IP Multimedia Subsystem) or SDP (Service Delivery Platform) by considering the technical patterns (protocols, reference points…) behind them. Then, we will study how to compose automatically a service in a web environment, according to a sequence of actions.

## References

1. North American Industry Classification System 2007 (Naics), US Dept. of Commerce (September 2007)
2. Global Employment Trends: January 2008, International Labour Office (2008)
3. Chesbrough, H., Spohrer, J.: A research manifesto for services science. Commun. ACM 49(7), 35–40 (2006)
4. Spohrer, J., Vargo, S.L., Caswell, N., Maglio, P.P.: The Service System Is the Basic Abstraction of Service Science. In: Proceedings of the 41st Annual Hawaii international Conference on System Sciences. HICSS, January 07 - 10, 2008, p. 104. IEEE Computer Society, Washington (2008)
5. Child, J., McGrath, R.G.: Organizations unfettered: Organizational form in an information-intensive economy. Aced. Manaement Journal 44(6), 1135–1148 (2001)

6. Zeithaml, V., Bitner, M., Gremler, D.: Services Marketing: Integrating Customer Focus Across the Firm, 4th edn. McGraw-Hill, New York (2006)
7. Shah, Denish, Rust, Roland, T., Parasuraman, A., Staelin, Richard, Day, G.S.: The Path to Customer Centricity. Journal of Service Research 9, 113–124 (2006)
8. Day, G.S.: Aligning the Organization with the Market. MIT Sloan Management Review 48(1), 41–49 (Fall, 2006)
9. Kellogg, K.C., Orlikowski, W.J., Yates, J.: Life in the Trading Zone: Structuring Coordination Across Boundaries in Postbureaucratic Organizations. Organization Science 17(1), 22–44 (2006)
10. Rust, R.T., Miu, C.: What academic research tells us about service. Commun. ACM 49(7), 49–54 (2006)
11. Tien, J.M., Berg, D.: A Case for Service Systems Engineering. J. Systems Science and Systems Eng., 113–128 (March 2003)
12. Spohrer, J., Maglio, P.P., Bailey, J., Gruhl, D.: Steps Toward a Science of Service Systems. Computer 40(1), 71–77 (2007)
13. `http://www.research.ibm.com/ssme/`
14. Gronroos, C.: In Search of a New Logic for Marketing: Foundations of Contemporary Theory. John Wiley & Sons Inc., Chichester (2007)
15. Pinhanez, C.: Service Systems as Customer-Intensive Systems and Its Implications for Service Science and Engineering. In: Proceedings of the 41st Annual Hawaii international Conference on System Sciences. HICSS, January 07 - 10, 2008, p. 117. IEEE Computer Society, Washington (2008)
16. Simoni, N.: Sous la direction de, Des réseaux intelligents à la nouvelle génération de services, Lavoisier (February 2007)
17. `http://www.openmobilealliance.org/`
18. `http://www.ist-spice.org`
19. Keck, D.O., Kuehn, P.J.: The Feature and Service Interaction Problem in Telecommunications Systems: A Survey. IEEE Transactions on Software Engineering 24(10), 779–796 (1998)
20. Bertin, E., Ben Yahia, I., Crespi, N.: Modeling IMS Services. Journal of Mobile Multimedia 3(2), 150–167 (2007)
21. Carugi, M., Hirschman, B., Narita, A.: Introduction to the ITU-T NGN focus group release 1: target environment, services, and capabilities. IEEE Communication Magazine 43(10), 42–48 (2005)
22. OMA, OMA Service Environment, Approved Version 1.0.4, 01, OMA-AD-Service-Environment-V1_0_4-20070201-A (February 2007)
23. 3GPP, Presence service using the IP Multimedia (IM) Core Network (CN) subsystem; TS 24.141, version 7.4.0 (September 2007)
24. 3GPP, Messaging using the IP Multimedia (IM) Core Network (CN) subsystem; TS 24.247, version 7.2.0 (June 2007)
25. 3GPP, Conferencing using the IP Multimedia (IM) Core Network (CN) subsystem, TS 24.147, version 7.6.0 (September 2007)
26. Rouse, W.B., Baba, M.L.: Enterprise transformation. Commun. ACM 49(7), 66–72 (2006)
27. Bertin, E., Fodil, I., Crespi, N.: A business view for NGN service usage. In: 2nd IEEE/IFIP International Workshop on Broadband Convergence Networks, 2007. BcN 2007, pp. 1–5, May 21 (2007)
28. Lamb, Kling: Reconceptualizing Users as Social Actors in Information Systems Research. Management Information Systems Quarterly 27(1), Article 2

29. Bertin, E., Lesieur, P.: Which architecture for integrated services? In: ICNS 2006. International conference on Networking and Services, p. 62 (2006)
30. Zhixiong, J., Leqiu, Q., Xin, P.: A Formal Framework for Description of Semantic Web Services. In: Proceedings of the 7th IEEE international Conference on Computer and information Technology. CIT, October 16 - 19, 2007, pp. 1065–1070. IEEE Computer Society, Washington (2007)
31. Arroyo, S., Lopez-Cobo, J.M.: Describing web services with semantic metadata. Int. J. Metadata Semant. Ontologies 1(1), 76–82 (2006)
32. Bedini, I., Gardarin, G., Nguyen, B.: Deriving Ontologies from XML Schema. In: Proceedings 4émes Journées francophones sur les Entrepôts de Données et l'Analyse en ligne (EDA 2008), Toulouse, France, June 5 - 6 (2008)
33. Patel-Schneider, P.F., Horrocks, I.: A comparison of two modelling paradigms in the Semantic Web. Web Semant. 5(4), 240–250 (2007)
34. Cai, H., Chung, J., Su, H.: Relooking at Services Science and Services Innovation. In: Proceedings of the IEEE international Conference on E-Business Engineering. ICEBE, October 24 - 26, 2007, pp. 427–432. IEEE Computer Society, Washington (2007)
35. Searle, J.R.: What is an institution? Journal of Institutional Economics 1(01), 1–22 (2005)

# Monitoring Web Services: A Database Approach*

Mohamed Amine Baazizi[1], Samir Sebahi[1], Mohand-Said Hacid[1], Salima Benbernou[1], and Mike Papazoglou[2]

[1] University Claude Bernard Lyon 1, LIRIS CNRS UMR 5205, France
[2] Tilburg University, The Netherlands

**Abstract.** Monitoring web services allows to analyze and verify some desired properties that services should exhibit. Such properties can be revealed by analyzing the execution of the services. Specifying monitoring expressions and extracting relevant information to perform monitoring is however not an easy task when the processes are specified by means of BPEL. In this paper we design a monitoring approach that makes use of business protocols as an abstraction of business processes specified by means of BPEL. High level queries are expressed against this abstraction and then translated into SQL queries that are evaluated against a database that stores the excustion traces of the services.

## 1 Introduction

The task of observing some process and tracking specific situations is known as monitoring. Amongst the many fields that witness special need of monitoring, there are Service Based Systems (SBS) which tend to support most today's applications. Today's enterprises rely on SBS to export their products. Different stakeholders may communicate with each other combining their existing products to constitute other ones more tangible to end users. This interaction is specified in a complex manner and advocates defining all the activities taking place in it. Moreover, each participant operates in an information system different from those used by the other parties. This was overcome by establishing a stack of protocols leveraging the heterogeneity that could take place. However, planing monitoring for those systems remains difficult since it advocates not only knowing details of the process being monitored, but also mastering tools and languages that served to specify it. All this to say that a new way to monitor business processes is more than required. It should take into account the difficulties of finding suitable models that bridge the gap between the modeled processes and the restricted knowledge of decision making actors.

In this paper, we build on previous work by Wombacher *et al.* [5] and Benatallah *et al.* [4] to provide a methodology for monitoring web services by considering their business protocols. The methodology is shown on figure 1. A BPEL specification is transformed into a business protocol. From the business protocol we

* The research leading to this result has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement 215483 (S-CUBE).

© Springer-Verlag Berlin Heidelberg 2008

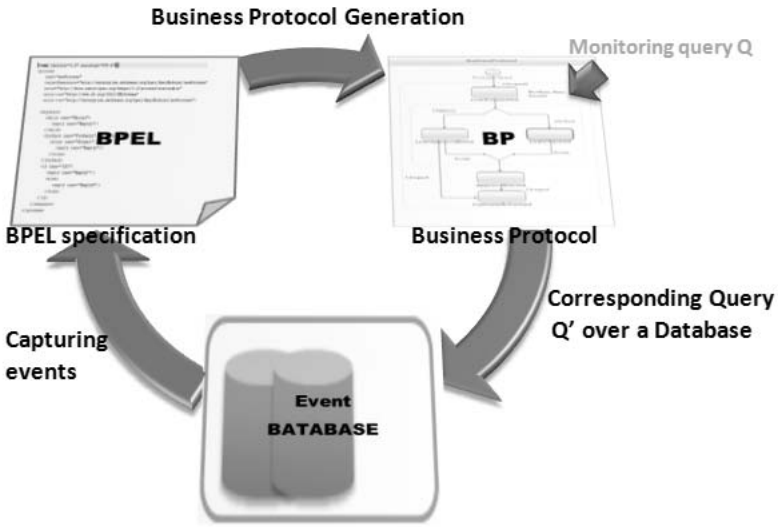


**Fig. 1.** The monitoring Framework

generate a (relational) database schema. Monitoring queries are specified against the protocols and translated into SQL queries against the relational database.

The paper is organized as follows: we first introduce, in section 2, a few notions on web services that are relevant in our work. Section 3 defines the monitoring process in web services. Section 4 describes our architectural and design principles of our approach for monitoring web services. We conclude in section 5 by summarizing our work and anticipating on necessary extensions.

## 2 Preliminaries

In this section we introduce relevant concepts to our work. We assume the reader familiar with Service Oriented Architectures (SOA) and notions related to Web services.

### 2.1 Business Process Execution Language (BPEL)

A business process consists of a bounded set of activities where data is manipulated and results are produced following the logic of the business it describes. Before the advent of web services, enterprises workflows were broadly used in describing the collaboration of many actors to produce a result. This was modeled by a graphical notation that shows the activities to be performed and the scheduling to respect in addition to the intermediate products that are passed between activities. In the same way, web services collaboration is captured by Business Process Execution Language (BPEL) [1], an XML-based standard used to orchestrate the enactment of the different services of the collaboration that interact to perform some specified task. It specifies the process behavior by defining the activities it is composed of and the external processes that interact with it.

### 2.2 Business Protocols

Making several partners' processes collaborate in an effective way needs an a priori look on their descriptions to depict eventual mismatches before their enactment. The actual standard for specifying web service offers neither enough visibility on the processes it specifies nor suitable tools that could help designers statically analyze the behavior of these processes that they manage to make communicate in a correct manner. This is why the authors in [4] investigated a way to represent the protocol that two services must follow to interact correctly. This was called business protocol since it represents the allowed conversations between a requester and a provider in terms of messages according to the states that they reached in the local execution of their respective business processes.

## 3 Monitoring Web Services

Web services are characterized by the fact that they are contracted somewhere in the time and may not be available after that or can still be available but in a different version making their evolution highly volatile. Additionally, every participant is mandated to correctly perform the task it has to carry out otherwise it will affect the entire process execution.

Monitoring copes with those deficiencies by observing web services execution after they have been deployed. It consists of a dedicated activity responsible of raising alert or triggering predefined actions when some situation is observed. It also consists of gathering useful information that will serve analysis. It could be extended with capabilities that allow avoiding some unwanted situations.

Monitoring web services was influenced by many techniques dealing with contracts and agreements, distributed systems property and safety verification, event processing, etc.

Many criteria could be considered when classifying monitoring approaches. According to [2], we can focus on the technique used to perform monitoring (verification, planning...) as well as on the data of interest to be monitored and many other aspects such as the abstraction of the language that serves monitoring specification and the degree of invasiveness on the monitored process.

## 4 A Database Approach for Monitoring Web Services

In this section we define the framework we are designing for monitoring business processes specified in BPEL. We will detail each component's functionality and the transformations undergone. We also provide the language used for formulating monitoring queries and characterize them regarding the abstraction upon which they are expressed.

### 4.1 The Overall Architecture

Figure 1 depicts the main components of the framework and the transformations that lead to each of them. We consider the executable BPEL specification of the business process to monitor. This specification will be mapped to a corresponding business protocol, provided some changes that will be discussed later. The mapping operation rests on a set of required transformation rules. A query language is then used for retrieving information by navigating through the states of the business process. Each query will be transformed into a suitable SQL query over a database which schema is a faithful mapping of the business protocol resulting from the transformation of the business process. This database is populated during the execution of the service supporting the business process.

### 4.2 A Business Protocol as an Abstraction

The abstraction of BPEL that we consider is a business protocols defined in [4] that we extend with variables associated with the states. The core definitions are kept identical. A protocol is defined as a tuple A=(Q, $q_0$, F, $\phi$ , $\Sigma$ ,$\psi$, Var) where:

– Q is a finite set of states the process goes through during its execution
– $q_0$ is the initial state
– F$\subseteq$ Q is the set of final states where F$\neq \emptyset$
– $\phi$ is the set of messages. There are two types of messages, those consumed by the protocol, these are assigned the polarity sign + and those produced by the protocol are assigned the - sign.
– $\Sigma \subseteq Q\times\phi\times Q$ is the transition set where every transition is labeled with a message name and its polarity.
– $\psi$ is a partial function that assigns to the states where a transition labeled with a receive message enters, the variable that is modified by this message. Not all states are assigned variables since only entering messages deliver information that is recorded in their corresponding variables.
– Var is the finite set of variables of the business process to be transformed.

### 4.3 Transformation of BPEL Business Processes to Business Protocols

In this section, we are interested in the mechanism that allows to generate an abstraction of a business process specified in BPEL by a set of rules. First, we have to define the different elements of a BPEL specification as stated in its specification [1].

For transformation purpose, we proceed by generating segments of the protocol corresponding to the basic activities and then combine the resulting segments by looking into the structured activities to which they belong.

#### 4.3.1 Transformation of Basic Activities

For each activity represented in BPEL syntax, we give its corresponding protocol segment definition. States named as q indexed with an integer i are just used for representation and could be renamed.

*invoke activity*
<INVOKE PARTNERLINK="PL" PORTTYPE="PT" OPERATION="OP" INPUTVARIABLE="INVAR" OUTPUTVARIABLE="OUTVAR"/>

is mapped into the following segment of the protocol

$(\{q_i, q_{i+1}, q_{i+2}\}, q_i, \{q_{i+2}\},\{\mathrm{m,n}\},\{(q_i, (-)m, q_{i+1}),(q_{i+1}, (+)n, q_{i+2})\}, \psi(q_{i+1}) = inVar, \psi(q_{i+2}) = outVar, \{inVar, outVar\})$

Here, $q_{i+1}$ is an intermediate state meaning that a message has been sent from a process to one of its partners and is blocked waiting for a message to be returned to change its state and affects the variable defined in this state.

*receive activity*
The receive activity which waits for a message that will be consumed takes the form <RECEIVE PARTNERLINK="PL" PORTTYPE="PT" OPERATION="OP" VARIABLE="VAR">

and is mapped to the protocol defined by

$(\{q_i, q_{i+1}\}, q_i, \{q_{i+1}\},\{\mathrm{n}\}, \{(q_i, (+)n, q_{i+1})\}, \psi(q_{i+1}) = Var, \{Var\})$

*assign activity*
<ASSIGN><COPY> <FROM>...</FROM> <TO VARIABLE="VAR"…/> </COPY> <ASSIGN/> is mapped to its corresponding protocol

$(\{q_i, q_{i+1}\}, q_i, \{q_{i+1}\},\{\},(q_i, Assign, q_{i+1}), \psi(q_{i+1}) = Var, \{Var\})$

The assign activity is local to a process and does not require any message exchange. This is why no polarity sign is used.

*reply activity*
<REPLY PARTNERLINK="PL" PORTTYPE="PT" OPERATION="OP" VARIABLE="VAR"> is mapped to the protocol

$(\{q_i, q_{i+1}\}, q_i, \{q_{i+1}\},\{\mathrm{m}\}, \{(q_i, (-)\mathrm{m}, \mathrm{q_{i+1}})\}, \psi(\mathrm{q_{i+1}}) = \mathrm{Var}, \{\mathrm{Var}\})$

Other activities like wait, exit, empty, throw and rethrow are available in the BPEL specification but not all are relevant. Wait which makes the process wait for a precise moment or until a certain time could be mapped to a business protocol using temporal transitions defined in [3] that are implicit transitions to be taken when the time constraint defined for them is satisfied. Exit is mapped to a transition leading to a final state.

### 4.3.2 Transformation of Structured Activities

Structured activities are used to link between basic activities following a logic we have in mind at design-time. This is done using different constructs such as **flow** which expresses that the activities defined in its scope run concurrently, **sequence** which links between basic or structured activities that are designed to run sequentially, **if-then-else** express conditional branching to a point in the process, **while** and **repeat-until** are used to loop through a set of activities and **pick** waits for a suitable message to trigger the corresponding action or a default action if time overruns. As done for the basic activities, we assign for each type of activity given in BPEL syntax its corresponding automaton definition.

## 4.4 A Monitoring Query Language

The monitoring methodology we propose consists of querying the business protocol corresponding to the business process we want to monitor rather than handling this latter itself. This is why we define our monitoring language upon business protocols to take advantage of the abstraction they offer. A business protocol represents the modeled system as a finite state automaton which transitions are annotated with messages exchanged and states are the mapping of the steps the process goes through until it ends. This visual representation of a system greatly simplifies its comprehension, and could hence be exploited to express queries in a natural and efficient manner. Figure 2 shows the business protocol of loan process system obtained from the transformation of BPEL code provided with the specification [1] using the transformation rules stated in § 4.3. The process starts by receiving customers' requests and decides, based on the asked amount, whether to check the loan request by the assessor service whose

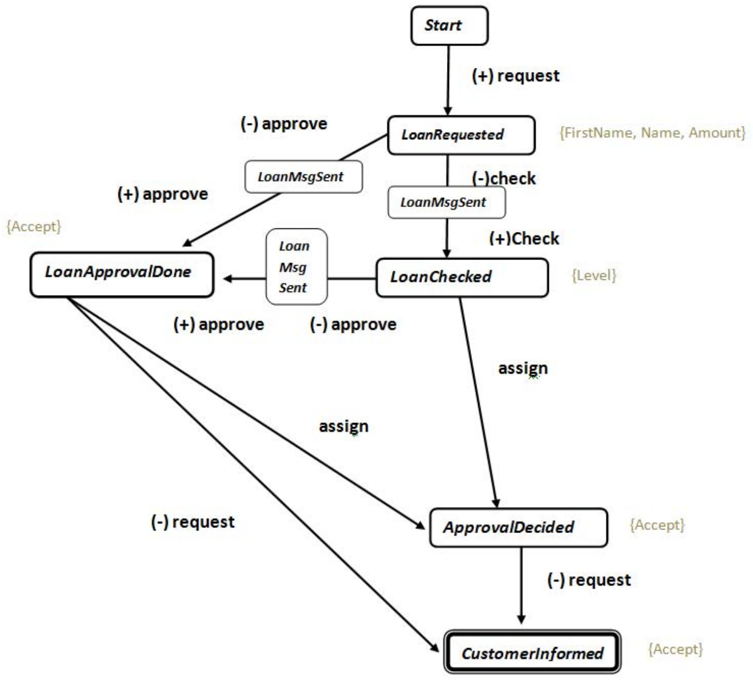


**Fig. 2.** The business protocol for the Loan approval example

role is to evaluate the risk of accepting the loan (expressed by the (-)Check message) or to check it directly by the approval service (expressed by the (-)Approve Message). If the risk is low, the acceptance will be decided locally and then assigned to a local variable that will be transmitted to the customer. Otherwise a processing from the approval service is needed and this latter has the responsibility to directly inform the customer in case of refusal or to return the response to the loan service that will forward it to the customer in case of acceptance. In both cases the customer is informed of the result of her/his request.

We first give some definitions that will serve introducing our monitoring language, then we will provide a syntax.

### 4.4.1 Execution Paths

As defined in the work [4] all traces left by the execution of a business process are captured by the corresponding business protocol. In the above example, the sequence **Start, request(+), LoanRequested, check(-), LoanMsgSent, check(+)LoanChecked** is an execution path. A complete execution path is an execution path that starts with the first state of the protocol and ends with its final state. It denotes a complete execution of the process represented by this protocol.

**Definition 1.** *Given a business protocol P, an execution path is formed by all the nodes (states of P) and edges (transitions of P) traversed during an execution. All the instances of the execution of one process generate execution paths that will be represented in a tree of executions. Figure 3 represents four paths of four different instances identified by their instance ID.*

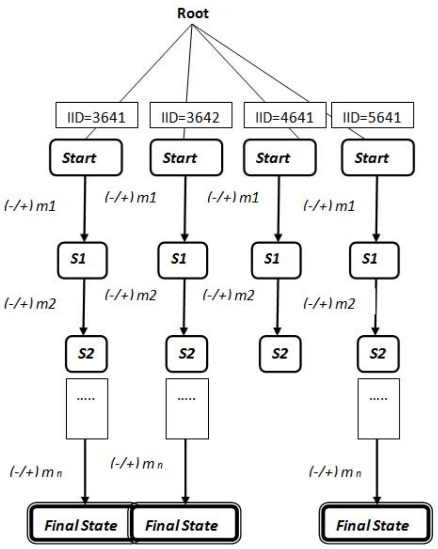


**Fig. 3.** A tree of all execution instances

**Definition 2.** *Let us consider a business protocol P defined as a tuple P=(Q, $q_0$, F, $\phi$, $\Sigma$,$\psi$, Var). A query over a business protocol P is a function that takes a path expression as input, that is a start node, an end node and eventually a set intermediate nodes constrained with the names of states defined in P (that is Q). It returns the values of variables defined on those nodes, an aggregation of those variables or a number of paths.*

### 4.4.2 The Query Language Syntax

A query over the business protocol P is an expression built using the the syntax shown figure 4, where terminals constitute keywords of the language and non-terminals are used in production rules and are thus underlined.

A query is composed of three clauses:

– Retrieve
– Where
– Constrain

The **Retrieve** clause specifies the information that will constitute the answer. It could be an attribute or a set of attributes. It could also be an aggregate result on the number of selected paths or the average number of executions leading to the selected paths.

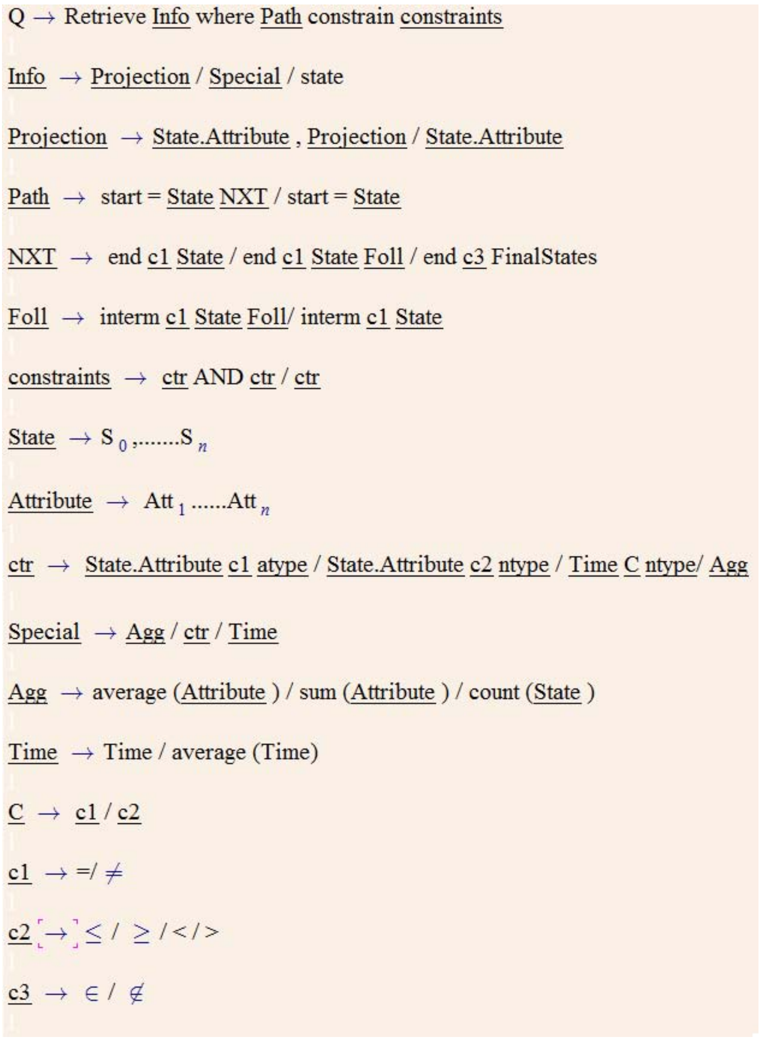
The **where** clause specifies the paths to select given a start node and an end node (the start and the end keyword respectively). We could restrict the selected paths by indicating intermediate nodes to cross or not to cross. The answer returned by the Retrieve clause is the set of attribute values of the selected paths if the query is intended to return attribute values, or an aggregate on these values or the number of paths that were selected. We can restrict even more the paths that will be selected using the **constrain** clause by fixing the values of attributes or the value of aggregates made on attributes values, or aggregates of time.

## 4.5 Query Evaluation

As mentioned previously, the queries formulated over the business protocol will be translated into SQL queries over an event database that captures the business process execution. First, we give the schema of such a database that will enable to retrieve the information as stated in the query language. Then, the above queries will be translated into their corresponding SQL queries over the database.

### 4.5.1 The Database Schema

The schema of the database is obtained by mapping each state of the business protocol to a relation of the database. Each relation is given the name of the state from which it is generated and the attributes identified in that state. Additional transformations are however required in case the variables defined in the business protocol (taken directly from BPEL specification which is XML-based) do not fit into relational table columns unless the host RDBMS allows storing such XML types.



Q → Retrieve Info where Path constrain constraints

Info → Projection / Special / state

Projection → State.Attribute , Projection / State.Attribute

Path → start = State NXT / start = State

NXT → end c1 State / end c1 State Foll / end c3 FinalStates

Foll → interm c1 State Foll/ interm c1 State

constraints → ctr AND ctr / ctr

State → $S_0$,.......$S_n$

Attribute → $Att_1$ ......$Att_n$

ctr → State.Attribute c1 atype / State.Attribute c2 ntype / Time C ntype/ Agg

Special → Agg / ctr / Time

Agg → average (Attribute ) / sum (Attribute ) / count (State )

Time → Time / average (Time)

C → c1 / c2

c1 → =/ $\neq$

c2 → $\leq$ / $\geq$ / < / >

c3 → $\in$ / $\notin$

**Fig. 4.** The query language syntax

Each state is designated by the ID it will have at run-time which is given by the BPEL engine to every running instance. At a given time, each state is linked with one and only one state (the following state in the execution path). The resulting table from a given state has the Instance ID (IID) as primary key, the variables of the state as attributes and a 1 to 1 multiplicity with the states coming right after it in the protocol representation.

For simplicity, we consider the example of figure 5 that shows the database schema resulting from the transformation of the protocol of figure 2. The intermediate states of the protocol (states without variables) are not mapped to any table in this schema. They are, however, stored elsewhere in a table called

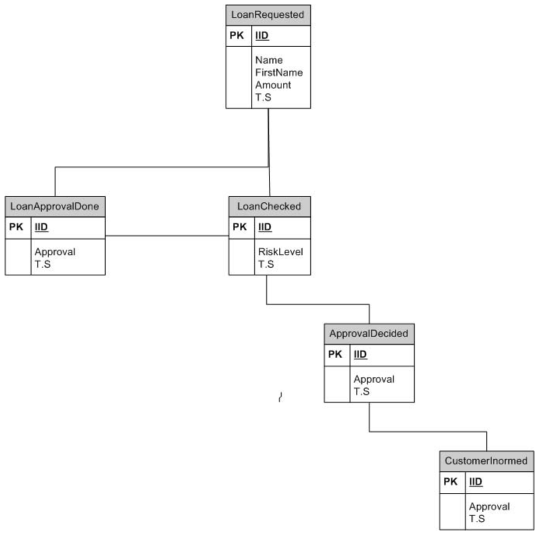


**Fig. 5.** A database schema of the protocol in figure 2

'Actual' that given an instance ID returns the name of the last state reached by the execution of an instance (the states in the business protocol). This table can be populated following two ways: each time a different instance is inserted into a table representing one state of the protocol, the 'state' attribute of the 'Actual' table is updated with the name of that table for the same IID. This is done by associating a trigger to every table. On each tuple insertion to a table, the name of this table is inserted into the state field of the 'Actual' table with teh schema:

```
Actual (IID,state, status,timestamp)
```

where: IID is the primary key and corresponds to the IID of the table where a tuple is inserted, state is the name of the table where the tuple is inserted, timestamp is the instance of time when the tuple was inserted into the original table and status has special significance which will be explained after. The trigger of a table resulting from a state $S_i$ can be defined as:

```
CREATE TRIGGER state_i_run
ON INSERT ON state_i_table
DECLARE
--X will hold the IID of the inserted tuple
X
BEGIN
-- If a tuple with the same IID already exists
IF X IN (SELECT IID FROM Actual) THEN
-- Update only the 'status' field
UPDATE Actual SET state='state_i'
ELSE
```

```
-- if the instance has not been yet recorded
INSERT INTO Actual (X,state_i)
END
```

Another way to populated the table is done at the level of the business process enactment by capturing messages sent in an invoke activity (cf. §4.3.1 ) that have not yet been responded by the partner link (if a response is required). The information which will be stored is the name of the partner link involved. Without this information we would never be capable of tracking the processes involved in failure or estimate their response time. Indeed, this prevents from mapping the intermediate states that denote in the business protocol that a message request has been sent and a response is expected.

At run-time, each created instance of the business process is stored in the database by filling the suitable fields with information generated during the execution. Each row of the database table is timestamped to enable the retrieval of temporal information.

The duration of a complete execution path is then given as the difference between its final and initial states' timestamps.

## 5 Conclusion

In this work, we provided a preliminary framework for business process monitoring using queries. This is just a starting work that will be helpful in:

- providing an abstraction of the monitored process that captures enough details relevant to monitoring issues, and not too much that could hinder the understanding of the modeled process;
- allowing an intuitive query formulation by visually selecting and eliminating parts of the process abstraction;
- ensuring efficient query evaluation by relying on relational databases that turn out to be more useful than expected when exploiting related mechanisms such as statistical analysis, actions triggering using the ECA paradigm but also off-line analysis since data is made persistent.

This work suggests reconsidering the problem of monitoring by taking another look that may lead to a solution when a important number of requirements will be satisfied. This is why we consider extending the high level query language so that it can deal with the maximum of situations one could need when monitoring any kind of process. This could be done by defining another syntax or extending the actual one while ensuring semantically correct queries with regards to a convention that will be made. A semantic compilation has to be defined at this level of abstraction so that high level queries will be mapped to the right SQL ones.

Since this monitoring works jointly with the BPEL standard specification, a deep review of the abilities of this latter could be of great benefit for optimization issues. for example, we could exploit the exception handling mechanisms defined in BPEL rather than redefining another one.

Additional extensions may concern querying flow activities after representing them and providing the suitable transformation mechanisms.

## References

1. Alves, A., Arkin, A., Askary, S., Barreto, C., Bloch, B., Curbera, F., Ford, M., Goland, Y., Guízar, A., Kartha, N., Liu, C.K., Khalaf, R., Koenig, D., Marin, M., Mehta, V., Thatte, S., Rijn, D., Yendluri, P., Yiu, A.: Web services business process execution language version 2.0 (OASIS standard). WS-BPEL TC OASIS (2007), http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html
2. Baresi, L., Di Nitto, E.: Test and Analysis of Web Services. Springer, Heidelberg (2007)
3. Benatallah, B., Casati, F., Ponge, J., Toumani, F.: On temporal abstractions of web service protocols. In: Belo, O., Eder, J., Cunha, J.F., Pastor, O. (eds.) CAiSE Short Paper Proceedings. CEUR Workshop Proceedings, vol. 161, CEUR-WS.org (2005)
4. Benatallah, B., Casati, F., Toumani, F.: Analysis and management of web service protocols. In: Atzeni, P., Chu, W.W., Lu, H., Zhou, S., Ling, T.W. (eds.) ER 2004. LNCS, vol. 3288, pp. 524–541. Springer, Heidelberg (2004)
5. Wombacher, A., Fankhauser, P., Neuhold, E.J.: Transforming BPEL into annotated deterministic finite state automata for service discovery. In: ICWS, pp. 316–323. IEEE Computer Society, Los Alamitos (2004)

# Milestones: Mythical Signals in UML to Analyze and Monitor Progress

Richard Torbjørn Sanders[1] and Øystein Haugen[1,2]

[1] SINTEF, N-7465 Trondheim, Norway
richard.sanders@sintef.no
[2] University of Oslo, Dept. of Informatics, N-0316 Oslo, Norway
oystein.haugen@sintef.no

**Abstract.** Many applications are evolving towards Service Oriented Architecture (SOA) with technologies such as Web services. Services can be modeled platform independently through UML2 collaborations in the upcoming UML profile for services, *SoaML*. We observe an increasing need for validation of services. However, such validation is often based on syntactic descriptions of the services and of their interfaces, which are insufficient to ensure that desired liveness properties are satisfied. In this paper, we present a language construct called "milestone" embedded in UML and define its semantics using mythical signals. We show how this interpretation of milestones can be used for liveness analysis and for runtime monitoring of services. The approach is illustrated with a simple bidding service.

## 1 Introduction

In recent years the software community has shown large interest in adopting Service Oriented Architectures (SOA) to overcome the challenges of distributed computing [1]. SOA is an architectural approach for constructing complex software-intensive systems from a set of interconnected and interdependent building blocks. A service is a stand-alone unit of functionality available through a formally defined interface.

While SOA in itself is not tied to any particular technology, most practitioners consider contemporary SOA to be that offered by web services. Semantic web services seek to characterize what a service can provide by offering means of expressing interfaces using Web Services Description Language (WSDL) [2]. Although WSDL aims at providing a formal definition of the interface to a service, it is restricted to a static description of operations and associated messages. This may change with the upcoming response to the OMG's RFP [3]. Called SoaML [4], the UML profile for services will allow one to formally define the behavior of a service on an interface, without binding the implementation to a particular technology. SoaML prescribes modeling services using UML2 Collaborations, see Fig. 1, as we argued in [5].

We have suggested the concept of *milestones* to express the desired behavior of a service [6]. In this article we show how the semantics of milestones can be defined by *mythical signals*, and how these are modeled by specialized UML *Comments* in SoaML. Mythical signals are signals which are useful for analysis and monitoring, but can be omitted in implemented systems. This work is a result of the SIMS project [7].

© Springer-Verlag Berlin Heidelberg 2008

The structure of this paper is as follows: first we present the rationale for milestones by way of a bidding example. Then we show how milestones are defined in UML, and how they contribute to the analysis and monitoring of progress. We also discuss related work, and finally conclude.

## 2 Buyers and Sellers – Progress of a Bidding Process

In this Section we introduce our illustrative example about buyers and sellers involved in a bidding process. We give an intuitive explanation of the example and then show how milestones can improve the understanding of the scenario as well as be a formal basis for liveness analysis.

Our situation is one where a seller offers an item to the market. We assume that the item is of considerable value such that a bidding process will be applied. The context is given by the UML collaboration shown in Fig. 1.

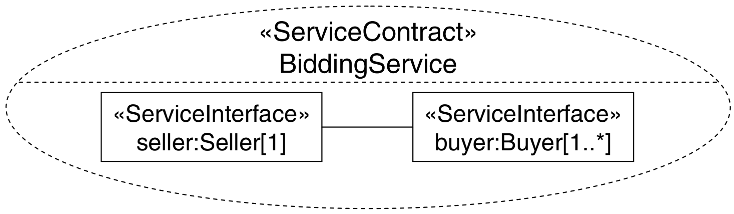


**Fig. 1.** Bidding service modeled as a collaboration

The seller will advertise the item for sale. We model this by assuming a message broadcast to a set of potential buyers. Some of the buyers will react favorably and return a message to indicate their interest. More information is then provided by the seller to all the buyers that have shown interest. After this preamble the bidding will start and a subset of the interested buyers will present a bid to the seller. We assume that the bid will contain additional information such as the price they are willing to pay, financing method etc. These additional pieces of information are not of much interest to our analysis and we have left them out of our simple model.

We then assume a series of bidding rounds where the seller will multicast to the remaining bidders the highest bid in the most recent round. Then the bidders may renew their bid with changed parameters. This procedure will go on for some time: it is not important for our analysis how many rounds or for how long the bidding process takes place.

Finally the bidding will terminate when the seller has selected a winner; the chosen one gets a message to pay and in return gets a contract for the item.

The bidding process is shown in Fig. 2. We have applied an augmented sequence diagram notation based on the notation and definition given in [8]. Compared to standard UML 2 sequence diagrams our notation has the capability to express broadcasting/multicasting in a precise yet compact way. The clue is that the buyer lifeline represents the whole set of buyers, but for each message (or combined fragment) we describe clearly what subset of the buyers will send or receive the message. Subsets of properties are defined in standard UML 2 [9].

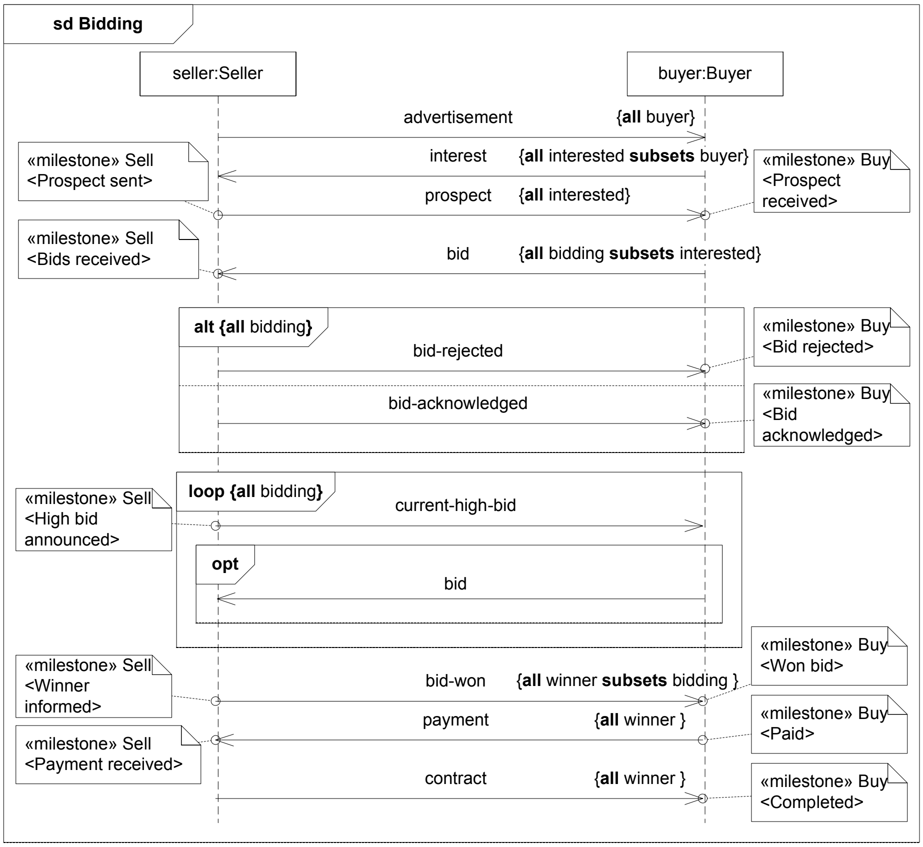

sd Bidding
seller:Seller
buyer:Buyer
advertisement
{all buyer}
interest
{all interested subsets buyer}
prospect
{all interested}
bid
{all bidding subsets interested}
«milestone» Sell <Prospect sent>
«milestone» Buy <Prospect received>
«milestone» Sell <Bids received>
alt {all bidding}
bid-rejected
bid-acknowledged
«milestone» Buy <Bid rejected>
«milestone» Buy <Bid acknowledged>
loop {all bidding}
current-high-bid
«milestone» Sell <High bid announced>
opt
bid
bid-won
{all winner subsets bidding }
«milestone» Buy <Won bid>
«milestone» Sell <Winner informed>
payment
{all winner }
«milestone» Buy <Paid>
«milestone» Sell <Payment received>
contract
{all winner }
«milestone» Buy <Completed>

**Fig. 2.** The bidding process modeled by a sequence diagram

Furthermore, we have added our notation for milestones that we shall introduce shortly. Milestones express that something useful has been achieved at this point in the behavior. For instance when a seller outputs “bid-won”, the bidding process has progressed to “Winner informed” in Fig. 2.

In a multi-stage interaction, like the bidding process, it makes sense to indicate a series of partial goals, each corresponding to something worthwhile having been achieved. In actual bidding interactions, many potential buyers never get past receiving prospects or having their bids rejected; these are nonetheless identifiable partial goals and represent fully acceptable outcomes of the interaction.

Likewise, it is useful to define goals for the participants of the interaction, in this case seller and buyer. The milestones are annotated such that the collaboration role to which they refer is made clear, i.e. `Sell` and `Buy`; as we shall shortly see, `Sell` and `Buy` are in fact progress signals. Fig. 2 states that the ultimate goal of the seller is to receive payment, while for the buyer it is to receive the contract – leading up to this are the steps or sub-goals of the bidding process needed to reach these final goals.

In addition to improving the reader's understanding, milestones are useful for stating requirements. For instance, in a bidding process buyers do not want prospects from sellers that withhold their acknowledgements until the buyers have lost interest. Nor do sellers want to reward buyers with a track record of withholding payment.

Most importantly, milestones act as a formal basis for liveness analysis. Liveness analysis is concerned with systems doing something good, and milestones can be used for expressing what is considered useful. As we shall see, with milestones we can analyze at design time how objects are capable of behaving, and/or monitor at runtime how objects actually behave. This can help us ascertain whether buyers and sellers are well-behaved and follow the intensions of a service specification. For instance, in a bidding process we do not want sellers that invariably reject all bids.

One benefit of milestones is that goal achievement is easier for people to recognize and follow, saving one from time-consuming analysis of programming code, procedure calls, message exchanges and other implementation artifacts. A benefit of this approach is that we do not need additional validation models unlike what is associated with formal methods; including milestones in a design provides analysis and monitoring opportunities without increasing the complexity of the model. At runtime, monitoring progress signals is a more practical instrument than monitoring all message exchanges and performing a progress analysis on these.

## 2.1 Defining Milestones in a UML Context

Milestones are marks of progress placed on behavioral elements of the UML specification. In our example given in Fig. 2 we have placed the milestones on message transmissions and message receptions. We may place milestones on any behavioral element where it is well defined when that behavioral element is executed at runtime.

The milestones in Fig. 2 are depicted as comments and the way they are written may lead people to believe that they represent pure constraints, but this would be a misconception. A constraint is something that is either true or false whenever the execution reaches this element. A milestone is something that states the fact that this behavioral element has been reached. While constraints are declarative and passive, milestones are imperative and active.

On the other hand, milestones share with constraints the fact that they are not necessary for the specification to execute properly. Just as all constraints can be removed from an executable model, so can all milestones. Both constraints and milestones are descriptions that are used for analysis alone. By analysis we mean not only the formal analysis provided by automatic means, but also informal analysis done by designers.

That milestones can be removed without changing the executable definition does not mean that milestones are useless or unimportant. In fact, the same can be said about other model elements; for instance sequence diagrams are normally considered redundant relative to the executable model. There are numerous algorithms that partially or totally generate executable models from sequence diagrams, but given a UML system defined with both state machines and sequence diagrams, the sequence diagrams will be used as advanced requirements on the executions and not the source of execution themselves.

Milestones are part of this tradition. They are also the first imperative constructs suggested in a UML context that have analysis as sole purpose. What should then happen when a milestone is encountered during execution? It is not sufficient to raise a flag since the same milestone may be encountered a number of times during an execution, and only raising a flag would not distinguish between encountering the milestone once and encountering it multiple times. The numbers or frequencies of these encounters may be of significance to what we call progress.

Thus, we decide that encountering a milestone should result in sending a signal to an observer totally outside our system. The signal name is given in the milestone along with an optional ordinal number representing the degree of progress. In our example in Fig. 2 we have used one progress signal `Sell` for the seller's progress and another progress signal `Buy` for the buyers' progresses.

A formal semantics for milestones would have to enhance the formal semantics of UML as such. The enhancement would have to comprise the external observer and a precise definition of exactly when during the execution of a behavioral primitive the progress signal should be sent. A formal semantics goes beyond this paper; here we explain in UML terms how the execution of milestones is.

The progress signals are declared as any other signal, and may in fact be signals that are used for other purposes in the specification. This means that the signals may have attributes and these attributes will get the runtime values at the time of the sending of the signal; the scope of the signal arguments follows normal UML scope rules.

The signals sent when milestones are encountered are sent only for the purpose of analysis and we imagine these signals are sent to a possibly fictitious observer outside our system. Since these signals could be omitted and since they may be understood as only being present for those that analyze, we call them "mythical signals". The term has some merit, as the term "mythical variable" was coined already in the seventies [10, 11]. The term "mythical variable" was used for variables that were not needed for the execution itself, but were auxiliary variables used to facilitate the reasoning. Typically the mythical variables have represented a history of states [12, 13] and as such they are similar to our mythical signals since the sequence of these signals represents a way to trace the history of the execution. In fact, we could apply a mythical variable to represent the sequence of mythical signals.

We have contributed the concept of milestones to the upcoming standard on service modeling (SoaML) [4], where the piece of the metamodel for milestones is depicted as in Fig. 3.

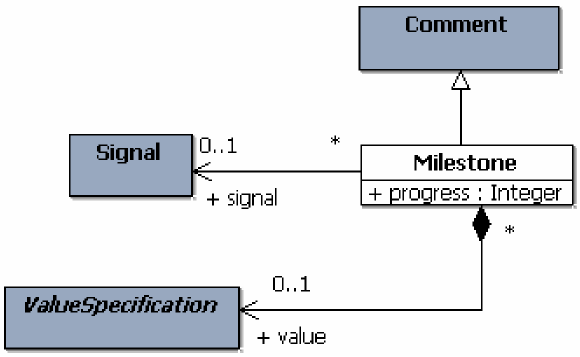


**Fig. 3.** Metamodel for Milestones

The metamodel expresses exactly what we have presented in this Section. A milestone is a kind of comment associated with a signal and an expression for the arguments of that associated signal. Furthermore, there is the progress value representing the degree of progress as an ordinal integer.

Milestones are not redundant in the sense that no other UML construct covers the same purpose. One may argue that sending a signal at selected places in the behavioral description can be done with ordinary UML means. This is true, but just including a number of signal-sending constructs does not serve the same purpose for the following reasons:

1. Milestones provide a uniform concept and notation across the different behavioral views of UML. Just sending signals will require different kinds of constructs for each of the behavioral diagrams.
2. Milestones are easily distinguished from sending signals that are necessary for the functioning of the system itself. This is what constitutes the mythical property of the milestones, that they are only used for analysis and not for specifying the functionality itself.
3. Milestones define sending of signals to an imaginary observer that is external to the outermost running system. Within UML there are constraints associated with sending of signals to indicate where the signal is sent. E.g. in sequence diagrams signal sending is represented by messages and these may go to the frame border. This is, however, the definition of a gate and must be matched where that Interaction is referenced. For our analytical purposes such constraints are counterproductive while they are practical and useful for pure functional purposes.

Milestones are not made superfluous by advanced model-driven debuggers either. It is possible to configure model execution support tools to report on reaching behavioral elements, but this is not well integrated with the modeling itself and it is an activity related to monitoring rather than analysis.

## 2.2 Progress Analysis

Validation at design time can be performed to ensure that components involved in a service will be able to interact safely with each other. We consider that components interact safely when their interactions do not lead to any unspecified signal receptions, deadlocks or improper termination.

The desired interface behavior of a participant in a service can be specified in a state machine like the one in Fig. 4 below. Here we see the specification of the interface behavior of the `Buyer` referred to in Fig. 1 and Fig. 2, where milestones are inserted at appropriate points. Interface behavior specifies the input and output signals on an interface, and is thus only a partial state machine; in particular it does not define causality of signal output. Fig. 4 constitutes what we have called a *semantic interface* [14]; it is indeed the milestones that contribute with the semantics of the interface.

Given a requirement specification detailing the interaction behavior, a component does not necessarily have to implement the complete behavior to be considered being compatible with the specification in terms of safety properties. Simply stated, a component can provide less output and accept more input than a specification, and (within

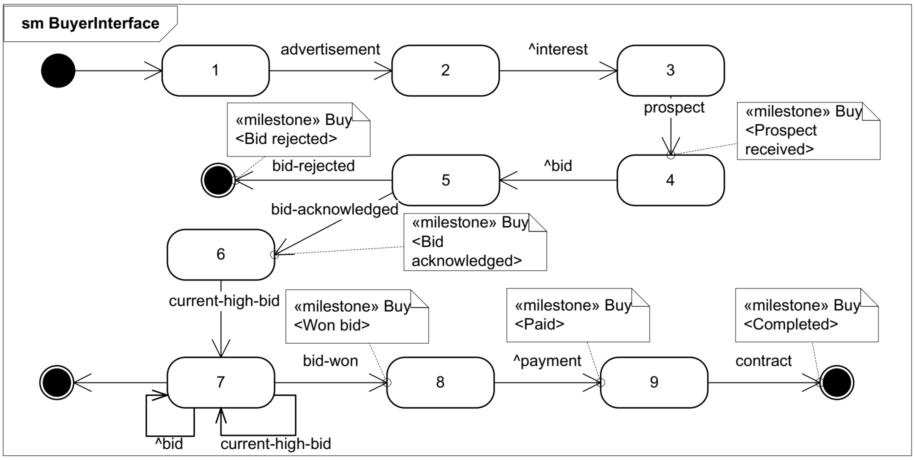


**Fig. 4.** Interface behavior of Buyer with milestones

certain constraints) still be safe, as we have discussed [14]. Interactions are considered safe if no unexpected signals are received and deadlocks do not arise (meaning that the peers wait endlessly for signals from each other). A safe behavior with less output is what we call a *safe subtype* [15].

However, that a component can interact safely in a service does not mean that it is useful. This is exemplified by the following diagram: Fig. 5 shows the possible behavior of a buyer that always responds with interest when it receives advertisements, but never does anything with the prospect it subsequently receives. The bidding world is full of would-be buyers that demonstrate behavior like this.

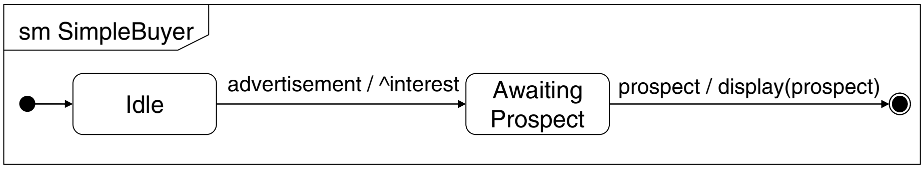


**Fig. 5.** Buyer behavior that is safe but not very useful

At first glance, Fig. 4 and Fig. 5 seem quite different; state names are different, and there are less states and signals in the latter. Some of these differences are due to the fact that the latter is the state machine of an object or classifier, while the former represents the interface behavior, and can be obtained by projection on an interface. Projection is due to the work of Floch [16], and is a mechanical process performed in order to simplify interface validation. In simple terms, projection removes events not visible on the interface, such as `display(prospect)` in Fig. 5, and transforms the state machine into a transition chart, e.g. the input of `advertisement` and the output of `interest` are placed in separate transitions, and auto-generated state names (numbers) are used. Projecting the state machine of `SimpleBuyer` (see Fig. 5) on the interface to the `Seller` results in the interface behavior in Fig. 6 below.

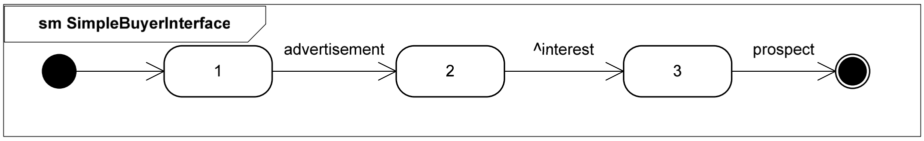


**Fig. 6.** Interface behavior of `SimpleBuyer`

Comparing Fig. 6 with Fig. 4 is straight-forward; we can see that `SimpleBuyer` performs the first part of the specification, ending where the specification reaches state `4`. Formally, a buyer acting according to the state machine in Fig. 5 can behave safely in the bidding service; it is capable of receiving the initial signals output from a seller[1], and does not output anything that a seller is incapable of handling according to Fig. 4. It is indeed a safe subtype, implying that it behaves safely in a bidding process. A seller would not receive any unexpected signals from such a buyer, nor would a seller wait endlessly for any signals from it, since bidding is not mandatory according to the service specification. However, seen from the perspective of a seller it is not very satisfactory, as such a buyer would never provide any bid.

This is where milestones come in. We can use milestones to analyze the behavior of an object or class and check if it is able to achieve the goals defined in a service specification. For buyers following the service behavior of the `SimpleBuyer` we see by comparing its projection in Fig. 6 with the specification in Fig. 4 that only one of the sub-goals can be achieved, `Buy <Prospect received>`, and neither the ultimate goal of seller nor buyer discussed earlier can be obtained in any interaction. Clearly, seen from the perspective of the seller, such buyer behavior is not fully satisfactory, since sellers want buyers to bid, not just to browse prospects. Analysis of the buyer behavior can disclose this; knowing this, sellers can take measures to avoid involving such participants in the bidding process.

A more serious case for the bidding process, however, would be buyers that win bids but are not able to provide payment, or sellers that never announce a winner, regardless of what bids are received. The latter case is an important one given the design of this bidding process: according to the specification, only the winner is informed, so active bidders are not able to check if a winner is ever announced. Analysis at design time can find this kind of discrepancy in the implemented behavior of a seller; monitoring at runtime, discussed below, can also reveal this.

Such design time validation exploits what is called a reachability analysis in formal methods. The example above is so trivial that no tool support is needed to perform the analysis. However, this is not the case in general; discovering errors and analyzing interaction behavior to find them can be difficult, and may require dedicated validation tools such as SPIN [17]. On the other hand, if one performs validation of collaborative behavior between components, one can simplify the analysis by focusing on interface behavior, and ensuring that the latter is well-formed (i.e. safe), meaning that nothing bad happens, and useful (i.e. live), meaning that it can achieve goals. As the example above shows, inserting milestones in interface behavior specifications can be

[1] We assume that a seller will not send messages like current-high-bid to buyers that do not submit bids. This is indeed as specified by the subset constructs in the sequence diagram.

used to validate liveness of subtypes, checking that progress can be achieved in the interactions. In the example above, the simple buyer is not able to achieve all the goals of the specification, and is thus not what we call a *live subtype* [15].

An example of more satisfactory buyer behavior is shown in Fig. 7 below. Using the validation approach mentioned above one can validate that the `SeriousBuyer` is fully goal compatible with that of `Buyer` in Fig. 4; analyzing its projection will show it to be a safe subtype, and that it contains transitions corresponding to all the milestones of the specification. This means it is a live subtype, and can achieve all the goals of the bidding process.

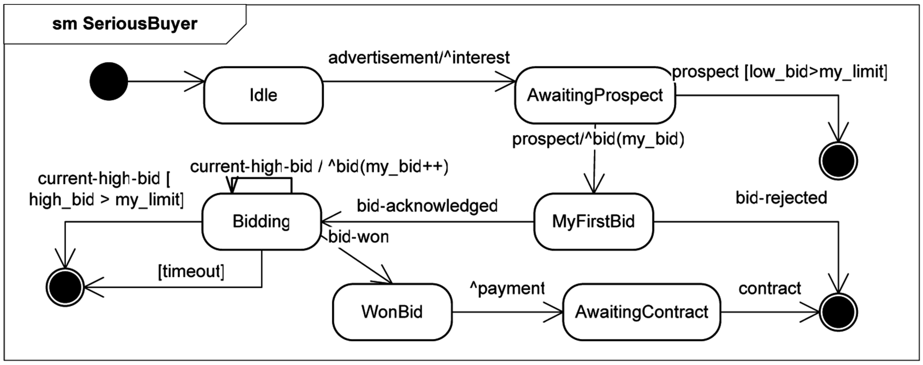


**Fig. 7.** A buyer that can achieve all the goals of the bidding service

The purpose of the analysis is to ascertain what progress is possible in interactions with a state machine. This does not imply that the goals are guaranteed to be fulfilled in every interaction. For instance, `SeriousBuyer` may be capable of achieving the ultimate goal of receiving a contract, but only if the bid is high enough. The analysis only shows that, given favorable conditions, this goal can be achieved. For the `SimpleBuyer`, however, the analysis concludes that no contract will ever be received.

In [15] we have suggested various kinds of milestones: graded milestones where a numeric value is specified by the label (for instance `<<Progress>> Buy (8)`), and service specific milestones of the kind used in Fig. 2 and Fig. 4. Both kinds are supported by the metamodel in Fig. 3. A graded milestone is modeled by the integer value, and can be used in making the best selection between a set of alternatives, for instance between a set of service implementations. A number of implementations may be compatible with a service specification, and a service discovery mechanism can select the implementation which exhibits the highest progress level.

Milestones are a mechanism that can be used for various needs; the analysis needs will determine what behavioral elements they are attached to. As can be seen from the metamodel in Fig. 3, milestones are specializations of comments; while comments can be attached to any model element in UML, milestones should only be attached to behavioral elements such as:

- MessageOccurrenceSpecifications in interactions (as exemplified in Fig. 2)
- Transitions in state machines (as exemplified in Fig. 4)
- ControlFlows in activities

For each of the different behavioral elements on which we may attach milestones we need to define precisely at what time instance at runtime the progress signal should be transmitted. For the three examples above, the MessageOccurrenceSpecification is not problematic as the associated event at runtime is normally considered to take zero time. The other two need more careful consideration. In most runtime situations transitions and control flows can also be considered instantaneous, but in cases where they are not we may define the progress as transmitted when the transition is finished or the control flow has given the control to the next activity node.

In the context of SoaML, service specifications seem natural candidates for exploitation of milestones; with this in place, service implementations can be validated with respect to their capabilities of fulfilling the goals expressed by the milestones.

Note that the examples presented here are simple, and do not demonstrate analysis of details such as guarded transitions. Furthermore, the analysis of interface behavior assumes that output eventually well be sent, which may not always be the case; validation of such properties using traditional state space exploration can be performed as a supplement - see [14, 15, 18] for further details of the validation approach. The benefit of milestone analysis lies primarily in the ease of use and understanding of the human designer, and the smaller size of the state space to be explored by machines.

### 2.3 Progress Monitoring

While we favor performing a comprehensive analysis of the models to establish progress and liveness, we also realize that most modelers do their analysis either through inspection or through testing. Formal analysis of milestones does not make testing obsolete. There may be characteristics of the system that are too difficult or too time consuming to analyze by symbolic means. Assume that there are strict time requirements on the bidding rounds, e.g. that a bidding round should not exceed one hour. A symbolic analysis of this requirement would require a lot of extra information about the behaviors of the bidders, and most certainly in a real situation the requirement could not be proved correct. Monitoring the progress on the other hand, requires only that the external observer (or in this case a "monitor") is actually implemented and the additional requirements on the progress checked by the implemented observer.

This monitoring could be compared with a special purpose debugging system or trace system. We could implement it as a state machine that consumes the mythical signals and reacts to them by compiling aggregate measures or performing checks on the fly.

In agile modeling one advocates small steps where every step is represented by an executable model. This is a very effective approach as long as it is easily established that the early immature systems perform what they should. Milestones and progress monitoring represent a lightweight approach to establishing that an immature system actually does something good without having to add all kinds of extra instrumentation to the model that must be removed later. The milestones may remain in the system, and the progress monitor may later choose not to react on certain progress signals.

## 3 Related Work

Clint [10] already in 1973 talks about "dummy statements" that cause "mythical" pushdown stacks to be updated with the new values of selected variables and thus recording the ongoing changes of the values. This is in fact quite similar to our approach of sending signals, only that he chose to keep the registration within the program. His aim was to prove correctness of co-routines and ours is to prove liveness of systems with concurrent, interacting processes.

In [11] Dahl applies mythical variables to count the number of times certain constructs are executed. This is again similar to our milestones as the mythical variable shows condensed information about the progress of the total program. Furthermore these mythical program variables are only meant for program analysis as their primary purpose is to appear in invariants that are used to prove the correctness of the program. [12] and [13] bring this technique one step further as the mythical variable is used to hold the whole history of the program.

The concept of milestones is inspired by mechanisms in traditional model checking, specifically the marking of so-called *progress states* in Promela [17]. While progress states markings are a mechanism used to detect non-progress cycles and livelocks in validation models, milestones are inserted into ordinary UML models in order to express, validate and monitor useful behavior, i.e. liveness in broad terms.

Milestones express the fulfillment of goals in interactions, and are a means of achieving automatic reasoning of goal achievement. The concept of goals is not unique to our work; for instance Business Motivation Model (BMM) defines the concepts of ends and goals [19]. In BMM, an *end* is something the business seeks to accomplish. An end does not include any indication of how it will be achieved. In BMM a *goal* is a statement about a state or condition of the enterprise to be brought about or sustained through appropriate means. The definitions of end and goal are not precise; the examples in [19] show normally only natural language. And although BMM goals can be formalized into OCL statements, this is less than what is desired; no algorithm can assess these goals, unlike the milestone approach we present here. Milestones seem to be a practical way of reasoning over goals.

## 4 Conclusion

In this article we have presented how the semantics of milestones is defined by so-called mythical signals, and how this concept can be included in extensions to UML such as the upcoming UML profile for services (SoaML). Milestones can be used to analyze and monitor service behavior, and (differently from model checking) do not require the construction of validation models; instead, milestones are embedded in ordinary UML models, to the benefit of the modeler.

We have discussed opportunities for such analysis and monitoring in terms of a simple bidding process. The application of milestones is not limited to toy examples; in ongoing research we are evaluating its use in mobile services [7].

## Acknowledgements

Our work is funded by the European Community under the Sixth Framework Programme, contract FP6-IST-027610 SIMS. The work is also partially supported by the MoSiS project ip06035, a project within the ITEA 2 – Eureka framework.

## References

1. Erl, T.: Service-Oriented Architecture - Concepts, Technology, and Design, 6th edn. Prentice Hall, Englewood Cliffs (2006)
2. W3C, Web Services Description Language (WSDL) Version 2.0 (2006), http://www.w3.org/TR/2006/WD-ws-cdl-10-primer-20060619/
3. OMG, UML Profile and Metamodel for Services (UPMS) RFP - soa/06-09-09 (2006), http://www.omg.org/cgi-bin/doc?soa/2006-9-9
4. OMG, Service oriented architecture Modeling Language (SoaML) - ad/2008-08-04 (2008), http://www.omg.org/cgi-bin/doc?ad/08-08-04.pdf
5. Sanders, R.T., et al.: Using UML 2.0 Collaborations for Compositional Service Specification. In: Briand, L.C., Williams, C. (eds.) MoDELS 2005. LNCS, vol. 3713, pp. 460–475. Springer, Heidelberg (2005)
6. Sanders, R.T., Floch, J., Bræk, R.: Dynamic Behaviour Arbitration using Role Negotiation. In: Next Generation Networks. Eunice 2003, Budapest, Hungary (2003)
7. SIMS - Semantic Interfaces for Mobile Services (2008), http://www.ist-sims.org
8. Haugen, Ø.: Challenges to UML 2 to describe FIPA Agent protocol. In: ATOP @ AAMOS 2008, Estoril, Portugal (2008)
9. OMG, UML 2.0 Superstructure Specification, Revised Final Adopted Specification, ptc/04-10-02, Object Management Group, Needham, MA, USA (2004)
10. Clint, M.: Program Proving: Coroutines. Acta Informatica 2, 50–63 (1973)
11. Dahl, O.-J.: An approach to Correctness Proofs of SemiCoroutines. In: Blikle, A. (ed.) MFCS 1974. LNCS, vol. 28, pp. 157–174. Springer, Heidelberg (1975)
12. Gjessing, S., Munthe-Kaas, E.: Trace Based Verification of Parallel Programs with Shared Variables. In: Twenty-Second Annual Hawaii International Conference on System Sciences, Kailua-Kona, HI, USA (1989)
13. Johnsen, E.B., Owe, O.: Object-Oriented Specification and Open Distributed Systems. In: Owe, O., Krogdahl, S., Lyche, T. (eds.) From Object-Orientation to Formal Methods. LNCS, vol. 2635. Springer, Heidelberg (2004)
14. Sanders, R.T., et al.: Service Discovery and Component Reuse with Semantic Interfaces. In: Prinz, A., Reed, R., Reed, J. (eds.) SDL 2005. LNCS, vol. 3530. Springer, Heidelberg (2005)
15. Sanders, R.T.: Collaborations, semantic interfaces and service goals: a way forward for service engineering, Norwegian University of Science and Technology (NTNU), Trondheim (2007), http://www.diva-portal.org/ntnu/abstract.xsql?dbid=1476
16. Floch, J.: Towards Plug-and-Play Services: Design and Validation using Roles, Norwegian University of Science and Technology (NTNU), Trondheim (2003)
17. Holzmann, G.J.: Design and Validation of Computer Protocols. Prentice Hall, Englewood Cliffs (1991)
18. SIMS deliverable D2.1 - Language and Method Guidelines, 1st version (2007), http://www.ist-sims.org/
19. OMG, Business Motivation Model (BMM) Specification dtc/07-08-03 (2007), http://www.omg.org/docs/dtc/07-08-03.pdf

# A Framework for Proactive Self-adaptation of Service-Based Applications Based on Online Testing[*]

Julia Hielscher[1], Raman Kazhamiakin[2], Andreas Metzger[1], and Marco Pistore[2]

[1] SSE, University of Duisburg-Essen, Schützenbahn 70, 45117 Essen, Germany
{hielscher,metzger}@sse.uni-due.de
[2] FBK-Irst, via Sommarive 18, 38050, Trento, Italy
{raman,pistore}@fbk.eu

**Abstract.** Service-based applications have to continuously and dynamically self-adapt in order to timely react to changes in their context, as well as to efficiently accommodate for deviations from their expected functionality or quality of service. Currently, self-adaptation is triggered by monitoring events. Yet, monitoring only observes changes or deviations after they have occurred. Therefore, self-adaptation based on monitoring is reactive and thus often comes too late, e.g., when changes or deviations already have led to undesired consequences. In this paper we present the PROSA framework, which aims to enable proactive self-adaptation. To this end, PROSA exploits online testing techniques to detect changes and deviations before they can lead to undesired consequences. This paper introduces and illustrates the key online testing activities needed to trigger proactive adaptation, and it discusses how those activities can be implemented by utilizing and extending existing testing and adaptation techniques.

## 1 Introduction

Service-based applications operate in highly dynamic and flexible contexts of continuously changing business relationships. The speed of adaptations is a key concern in such a dynamic context and thus there is no time for manual adaptations, which can be tedious and slow. Therefore, service-based applications need to be able to self-adapt in order to timely respond to changes in their context or their constituent services, as well as to compensate for deviations in functionality or quality of service. Such adaptations, for example, include changing the workflow (business process), the service composition or the service bindings.

In current implementations of service-based applications, monitoring events trigger the adaptation of an application. Yet, monitoring only observes changes or deviations *after* they have occurred. Such a reactive adaptation has several important drawbacks. First, executing faulty services or process fragments may have undesirable consequences, such as loss of money and unsatisfied users. Second, the execution of adaptation activities on the running application instances can considerably increase execution time, and therefore reduce the overall performance of the running application. Third,

[*] 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).

© Springer-Verlag Berlin Heidelberg 2008

it might take some time before problems in the service-based application lead to monitoring events that ultimately trigger the required adaptation. Thus, in some cases, the events might arrive so late that an adaptation of the application is not possible anymore, e.g., because the application has already terminated in an inconsistent state.

*Proactive adaptation* presents a solution to address these drawbacks, because – ideally – the system will detect the need for adaptation and will self-adapt before a deviation will occur during the actual operation of the service-based application and before such a deviation can lead to the above problems.

In this paper we introduce the *PROSA* framework for *PRO*-active *S*elf-*A*daptation. PROSA's novel contribution is to exploit online testing solutions to proactively trigger adaptations. Online testing means that testing activities are performed during the operation phase of service-based applications (in contrast to offline testing which is done during the design phase). Obviously, an online test can fail; e.g., because a faulty service instance has been invoked during the test. This points to a potential problem that the service-based application might face in the future of its operation; e.g., when the application invokes the faulty service instance. In such a case, PROSA will proactively trigger an adaptation to prevent undesired consequences.

The remainder of the paper is structured as follows: In Section 2 we give an overview of current research results on using monitoring to enable (reactive) adaptation and of the state-of-the-art in online and regression testing. In Section 3 we present the PROSA framework. While describing the key elements of the framework, we discuss how those could be implemented by utilizing or extending existing testing and adaptation techniques. Section 4 introduces several application scenarios to illustrate how PROSA addresses different kinds of deviations and changes. Finally, Section 5 critically reviews the framework and highlights future research issues.

## 2 State-of-the-Art

### 2.1 Monitoring for Adaptation

Existing approaches for adaptation of service-based applications rely on the possibility to identify and realize – at run-time – the necessity to change certain characteristics of an application. In order to achieve this, adaptation requests are explicitly associated to the relevant events and situations. *Adaptation requests* (also known as adaptation requirements or specifications) specify how the underlying application should be modified upon the occurrence of the associated event or situation. These events and situations may correspond to various kinds of failures (like application-level exceptions and infrastructure-level failures), changes in contextual settings (like execution environment and usage context), changes among available services and their characteristics, as well as variations of business-level properties (such as key performance indicators).

In order to detect these events and situations, the majority of adaptation approaches resorts to exploiting *monitoring* techniques and facilities, as they provide a way to collect and report relevant information about the execution and evolution of the application. Depending on the goal of a particular adaptation approach, different kinds of events are monitored and different techniques are used for this purpose.

In many approaches (e.g., [1,2,3,4]) the events that trigger the adaptation are failures. These failures include typical problems such as application exceptions, network problems and service unavailability [1,4], as well as the violation of expected properties and requirements. In the former case fault monitoring is provided by the underlying platform, while in the latter case specific facilities and tools are necessary. In [2] Baresi et al. define the expected properties in the form of WS-CoL assertions (pre-, post-conditions, invariants), which define constraints on the functional and quality of service (QoS) parameters of the composed process and its context. In [5] Spanoudakis et al. use properties in the form of complex behavioral requirements expressed in event calculus. In [3] Erradi at al. express expected properties as policies on the QoS parameters in the form of event-condition-action (ECA) rules. When a deviation from the expected QoS parameters is detected, the adaptation is initiated and the application is modified. In such a case, adaptation actions may include re-execution of a particular activity or a fragment of a composition, binding/replacement of a service, applying an alternative process, as well as re-discovering and re-composing services. In [6] Siljee et al. use monitoring to track and collect the information regarding a set of predefined QoS parameters (response time, failure rates, availability) infrastructure characteristics (load, bandwidth) and even context. The collected information is checked against expected values defined as functions of the above parameters, and in case of a deviation, the reconfiguration of the application is triggered.

Summarizing, all these works follow the reactive approach to adaptation, i.e., the modification of the application takes place *after* the critical event happened or a problem occurred.

The situation with reactive adaptation is even more critical for approaches that rely on post-mortem analysis of the application execution. A typical monitoring tool used in such approaches is the analysis of process logs [7,8,9]. Using the information about histories of application executions, it is possible to identify problems and non-optimalities of the current business process model and to find ways for improvement by adapting the service-based application. However, once this adaptation happens, many process instances might have already been executed in a "wrong" mode.

## 2.2 Online Testing and Regression Testing

The goal of testing is to systematically execute service instances or service-based applications (service compositions) in order to uncover failures, i.e., deviations of the actual functionality or quality of service from the expected one.

Existing approaches for testing service-based applications mostly focus on testing during design time, which is similar to testing of traditional software systems. There are a few approaches that point to the importance of online testing of service-based applications. In [10] Wang et al. stress the importance of online testing of web-based applications. The authors, furthermore, see monitoring information as a basis for online testing. Deussen et al. propose an online validation platform with an online testing component [11]. In [12] metamorphic online testing is proposed by Chan et al., which uses oracles created during offline testing for online testing. Bai et al. propose adaptive testing in [13,14], where tests are executed during the operation of the service-based application and can be adapted to changes of the application's environment or of the application itself. Finally, the role of monitoring and testing for validating service-based applications is examined in [15], where the authors propose to use both strategies in combination. However, all these approaches do not exploit testing results for (self-)adaptation.

An approach related to online testing is regression testing. Regression testing aims at checking whether changes of (parts of) a system negatively affect the existing functionality of that system. The typical process is to re-run previously executed test cases. Ruth et al. [16,17] as well as Di Penta et al. [18] propose regression test techniques for Web services. However, none of the techniques addresses how to use test results for the adaptation of service-based applications.

Summarizing, in spite of a number of approaches for online testing and regression testing, none of these approaches targets the problem of proactive adaptation. Still, several of the presented approaches provide baseline solutions that can be utilized and extended to realize online testing for proactive adaptation. This will be discussed in the following section.

## 3 PROSA: Online Testing for Proactive Self-adaptation

As introduced in Section 1, the novel contribution of the PROSA framework is to exploit online testing for proactive adaptation. Therefore, the PROSA framework prescribes the required online testing activities and how they lead to adaptation requests. Figure 1 provides an overview of the PROSA framework and how the proactive adaptation enabled by PROSA relates to "traditional" reactive adaptation which is enabled by monitoring.

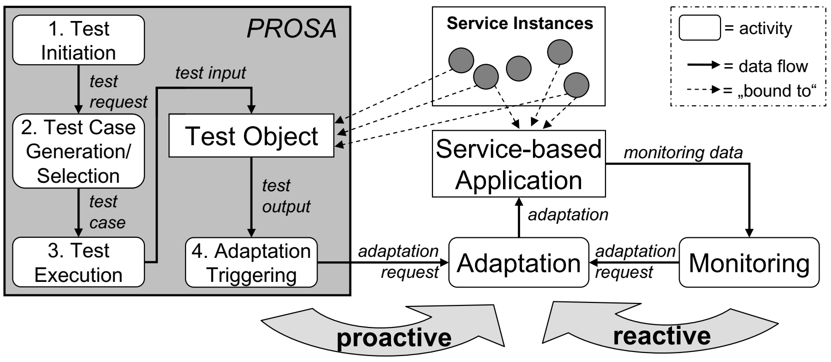


**Fig. 1.** The PROSA Framework

The PROSA framework prescribes the following four major activities:

1. *Test initiation*: The first activity in PROSA is to determine the need to initiate online tests during the operation of the service-based application. The decision on when to initiate the online tests depends on what kind of change or deviation should be uncovered (see Section 3.1).

2. *Test case generation/selection*: Once online testing has been initiated by activity 1, this second activity determines the test cases to be executed during online testing. This can require creating new test cases or selecting from already existing ones (see Section 3.2).
3. *Test execution*: The test cases from activity 2 are executed (see Section 3.3).
4. *Adaptation triggering*: Finally, an analysis of the test results provides information on whether to adapt the service-based application and thus to create adaptation requests (see Section 3.4).

It should be noted that – as depicted in Figure 1 – online testing does not interfere with the execution of the actual application in operation, i.e. with those instances of the application which are currently used by actual users. Rather, online testing performs tests of the constituent parts of the service-based application (e.g., individual services or service compositions) independent from and in parallel to the operating applications.

Details about the above activities and how those can be implemented with existing techniques are discussed in the remainder of this section.

## 3.1 Test Initiation

In order to initiate the actual online testing activities (PROSA's activities 2 and 3), two questions need to be answered: "When to test?" and "What to test?". The answer of these questions depends on the kinds of changes or deviations that should be *proactively* addressed in addition to reactive techniques like monitoring. Those possible kinds of changes are listed in Table 1.

To give an answer to the question "When to test?", Table 1 provides an explanation when to initiate online testing depending on the kind of change or deviation. Those kinds of changes and deviations are illustrated in more detail in Section 4, where different application scenarios for PROSA are introduced.

**Table 1.** Different cases for initiating online testing

| *Case* | *Why to initiate online testing?* | *When to initiate online testing?* | *What to test?* |
|---|---|---|---|
| *1* | Uncovering failures introduced due to the adaptation of the service-based application. | Once the respective adaptation (e.g. binding of a new service) has been performed. | service or composition |
| *2* | Detecting changes in the service-based application or its context that could lead to failures in the "future". | Once monitoring has detected a change that does not reactively trigger an adaptation. | service or composition |
| *3* | Identifying failures of an application execution. | Periodically (e.g. randomly or by testing future service invocations along the execution path of the application). | composition |
| *4* | Uncovering failures (i.e., deviations from expected functionality or quality) or unavailability of constituent services. | Periodically (e.g., randomly or by predicting future service invocations along the execution path of the application). | service |

To provide an answer to the question "What to test?" (i.e., to determine the test object), we have considered the following two major strategies that can be performed in order to uncover the different kinds of changes or deviations (Table 1 shows what strategy could be followed depending on the kind of change or deviation):

– *Testing constituent service instances:* Similar to unit or module testing, the individual, constituent service instances of a service-based application can be tested (i.e., the service instances that are or will be bound to the service-based application).
– *Testing service compositions:* Similar to system and integration testing, the complete service composition of a service-based application or parts thereof can be tested.

To implement activity 1 of PROSA, one can rely on information provided by existing monitoring techniques for case 2 (see Table 1) or adaptation techniques for case 1. The other cases require new and specific techniques, which can be very simple (like randomly triggering the tests) or more challenging (like predicting future service invocations along the execution path of the application).

## 3.2 Test Case Generation/Selection

In Section 3.1 two strategies for online testing were introduced. In order to implement these two different strategies and thus to realize activity 2 of the PROSA framework, different kinds of techniques for determining test cases have to be employed:

– *Testing constituent service instances:* For testing constituent service instances, existing techniques for test case generation from service descriptions, like WSDL, can be exploited (e.g., [19,20,21]). Additionally, test cases from the design phase can be re-used if such test cases exist. However, usually the test cases from the design phase will not suffice, because typically at that time not all services are known due to the adaptation of a service-based application that can happen during run-time.
– *Testing service compositions:* For testing service compositions, test cases can be generated from composition specifications, like BPEL (e.g., [22,23]). If a set of test cases for testing service compositions already exists, online testing has to determine which of those test cases to execute again (i.e., test cases have to be selected). This is similar to regression testing, which has been discussed in Section 2.2. Consequently, existing techniques for regression testing of services (like [16,17,18]) can be utilized.

A more detailed survey on existing test case generation and selection techniques for service-based applications can be found in [24].

## 3.3 Test Execution

The responsibility of activity 3 in the PROSA framework is to execute the test cases that have been determined by activity 2. This means that the test object (which is either a service instance or a service composition) is fed with concrete inputs (as defined in the test cases) and the produced outputs are observed.

The test execution can be implemented by resorting to existing test execution environments, e.g., the ones presented in [19,18]. It is important to note that invoking services can lead to certain "side effects" which should not occur when invoking the service for testing purposes only (this problem is also discussed in [22]). As an example, when invoking the service of an online book seller for testing purposes, one would not like to have the "ordered" books actually delivered. Thus, it is necessary to provide certain services with a dedicated test mode. As an example, one could follow the approaches suggested for testing software components, where components are provided with interfaces that allow the execution of the component in "normal mode" or in "test mode" (see [25]).

### 3.4 Adaptation Triggering

The final activity 4 of PROSA determines whether to issue an adaptation request, which ultimately leads to the modification of the service-based application. Such an adaptation request should be issued when the observed output of a test deviates from the expected output, i.e., whenever a test case fails. This includes deviations from the expected functionality as well as from the expected quality of service.

As has been discussed above, existing adaptation solutions rely on monitoring to issue adaptation requests whenever a deviation is observed (see reactive loop in Figure 1). In order to exploit those existing solutions (see Section 2.1), triggering of adaptations based on online testing should conform to the requests from the monitoring component. Thereby, activity 4 could be implemented within a unified adaptation framework.

To achieve such a unification, the following two issues need to be resolved: First, specific adaptation requests should be explicitly assigned to individual test cases. In reactive approaches such adaptation requests are assigned to certain monitoring events. The events may represent application or network failures (e.g., service is unavailable), violation of assertions (e.g., post-condition on data returned by service call) or even of complex behavioral properties (e.g., if flight is found but there are no rooms available, the trip plan can not be created). In a similar way, test cases represent dedicated execution scenarios, where specific deviations or changes can be checked (this has been highlighted in Table 1). If the test fails, this is similar to the occurrence of a monitoring event, and thus the adaptation assigned to the test case is triggered.

Second, it may be necessary to modify the adaptation requests from monitoring in order to take into account the specifics of proactive adaptation. Indeed, some adaptation requests from monitoring might specify instructions that are not applicable in proactive adaptation (e.g., "retry" operation, or "rollback to safe point"). Therefore, the specification should be changed such that these instructions do not appear when used for proactive adaptation. An interesting line of future work in these regards could be to devise means to automatically derive adaptation requests for proactive adaptation from the adaptation requests already available for monitoring.

## 4 Application Scenarios

In this section we illustrate how PROSA enables the proactive adaptation of a service-based application. For this purpose we introduce an example application based on which

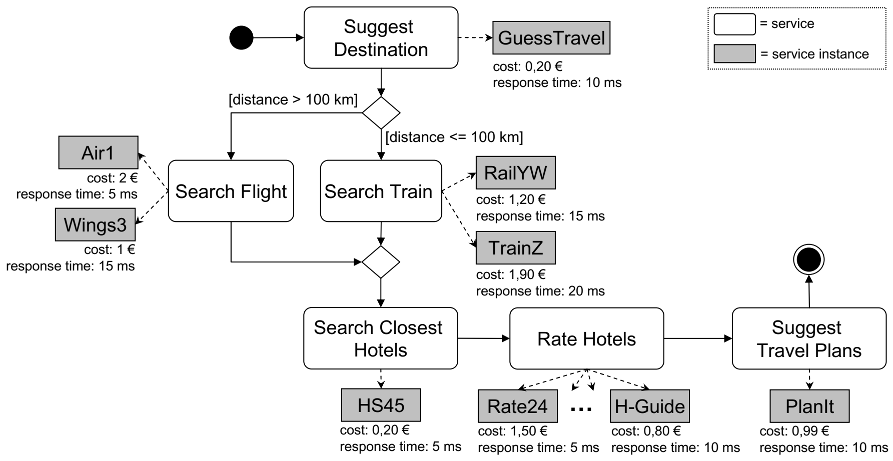


**Fig. 2.** Example Application: "Travel Planning"

we describe scenarios that demonstrate how PROSA can be applied to the different cases for online testing introduced in Table 1. The service composition of the example and possible constituent service instances are depicted in Figure 2.

Our example application provides a travel planning service, which includes a combined search for transportation and hotel accommodation. The constituent services of this application are invoked in the following order:

1. *Suggest destination:* First, the user of the application is provided with a suggestion of different travel destinations based on her/his preferences.
2. *Search flight/train:* Once the user has chosen a destination, the application will determine the best way to reach that destination. Depending on the distance to the suggested destination, either an appropriate flight or a train connection is searched.
3. *Search closest hotels:* After a suitable means of transportation has been found, hotels in the vicinity of the airport or the railway station of the destination are located.
4. *Rate hotels:* Using one of the many hotel rating services available, each hotel from the list is checked for its rating and the hotel list, sorted according to the rating, is returned.
5. *Suggest travel plans:* Finally, the first hotel from the sorted list (i.e., the one with the best overall rating) is chosen and the travel information (itineraries, information about the hotel, etc.) is compiled to produce a comprehensive travel plan.

In Figure 2, gray boxes denote concrete service instances that can be bound to the application in order to compute the travel plan. Some of those concrete service instances can already be known at design time, while others are dynamically discovered or added due to adaptations during the operation of the service-based application. The annotated information about cost and response time denotes the negotiated quality for each of the service instances (e.g., by means of service level agreements).

### 4.1 Case 1: Failure Introduced due to Adaptation

Let us assume that the service instance “H-Guide” was bound to our service-based application at operation time, because the service instance “Rate24” has turned out to be too expensive. The binding of that new service instance is reported by the adaptation component to the PROSA framework. Consequently, PROSA’s activity 1 triggers the online testing activities, which react to this adaptation by determining test cases to check whether the newly bound service instance behaves as expected (see Table 1, case 1). Let us say that the expected output of one of those test cases is “Palermo Premium Class Hotel”, which clearly is the hotel with the best ratings for the chosen location. Unfortunately, the observed output of “H-Guide” is “Casa Palermo”, which is the hotel with one of the lowest ratings (the reason for this presumed failure is that – other than expected – “H-Guide” returns the list of hotels in ascending order, starting with the lowest ratings). Online testing reports this failure to the adaptation component, which can – for example – switch back to the initial service instance “Rate24”, which has already been used successfully.

### 4.2 Case 2: Change That Could Lead to Failures in the Future

Let us assume that a new regulation concerning the pricing of flights enters into force during the operation of the service-based application. The regulation requires that the overall cost of a flight (including taxes) has to be stated and that it may not anymore be stated as the price for the flight with the note “plus taxes”. This legal change thus represents a change in the context of the application (see Table 1, case 2). As a result, PROSA will initiate online testing activities – when this new regulation enters into force – in order to determine whether the constituent service instances of the service-based application conform to this new regulation. This means that online tests will be triggered in order to check whether the service instances for flight booking (“Air1” and “Wings3”) conform to the new regulation. If one of those service instances does not implement the new regulation, the service-based application will be adapted accordingly before that service instance is invoked during the actual operation of the application.

### 4.3 Case 3: Failure of an Application Execution

The output of “search train” (resp.“search flight”) contains the name of the city close to the airport or the railway station. This city name is passed on to “search closest hotels” in order to determine the list of hotels in the vicinity of the destination. Let us assume that the service instance “RailYW” always provides the name of the destination in “short” form, meaning that even if there is more than one city with this name, like “Frankfurt am Main” and “Frankfurt an der Oder”, this service instance will always return “Frankfurt”. When the hotel searching service “HS24” receives such an ambiguous input, it will terminate with an error message. By running test cases to check deviations in the service composition (see Table 1, case 3), PROSA can uncover such a failure and – as a proactive corrective action – can request that a different service instance is bound to the application (e.g., “TrainZ”).

### 4.4 Cases 4: Failure of a Constituent Service

For the booking of an appropriate flight, two service instances are available: “Air1” and “Wings3”. “Air1” is used for premium clients, which are willing to pay more for a shorter response time. “Wings3” is the preferred choice of clients who want to save money. At operation time the online testing component runs several test cases per hour (periodically testing, see Table 1, case 4). Let us assume that one of those tests uncovers that “Wings3” does not respond. PROSA then provides the adaptation component with this information, such that the alternative service instance “Air1” (which is working as expected) is used for all queries.

## 5 Discussion and Perspectives

This paper has introduced the PROSA framework, which defines key activities for enabling the proactive self-adaptation of service-based applications. The novel contribution of PROSA is to exploit online testing techniques in order to anticipate future deviations or changes of a service-based application and thereby to trigger adaptation requests. In addition to the definition of those key activities, the paper has discussed how those activities can be implemented by building on or extending existing testing and adaptation techniques.

In contrast to the “traditional” form of reactive adaptation (e.g., based on monitoring), PROSA provides the following important benefits: First, changes or deviations from expected functionality or quality of service can be uncovered and addressed before they lead to undesirable consequences. Second, the execution of adaptation activities – if done proactively – does not interfere with the execution of the actual application instances, i.e., the users of the application won’t be affected by the adaptation. Third, proactive adaptation can provide adaptation requests early enough such that an adaptation of the service-based application still is possible (in contrast to reactive adaptation, where the application can have already terminated in an inconsistent state, for instance). Due to these benefits, we are confident that the PROSA framework will enable novel service-based applications that are able to proactively adapt and thus to better meet their expectations.

In addition to uncovering failures, monitoring is also often used to improve (or optimize) a service-based application. Accordingly, online testing could be used in this respect, for instance by determining the best possible alternative for an adaptation decision before the adaptation is executed. This means whenever an adaptation decision is imminent and different alternatives exist, those alternatives could be “pre-tested” and the best one chosen. For example, consider an adaptation specification, where on failure of a service instance three strategies are defined: retry invoking the service instance three times, replace the service instance with another service instance, change the service composition to use different services. Testing can now “simulate” all those three strategies and maybe detect that “change composition” is the only way to successfully drive the adaptation.

Although exploiting only testing for proactive adaptation provides many benefits, we acknowledge at this stage that further work is required in order to demonstrate the applicability of the PROSA idea in practice. One aspect that, for example, has to be investigated, is the possible impact of the execution of test cases on the performance of the application. Thus, key issues that we will target in our future work are to create a proof-of-concept prototypes based on existing techniques and tools (as discussed in the paper) and to apply these prototypes to realistic cases.

As we have briefly pointed out in the paper, proactive and reactive adaptation may work together in an integrated dynamic adaptation framework. In such a framework, online testing and monitoring could mutually benefit from each other, thereby improving the overall quality and efficiency of adaptation. In further work, we thus plan to investigate on how to best exploit the synergies between monitoring and testing. As an example, the results of monitoring may be used to identify "better" test cases for online testing. When complex behavioral properties are monitored (e.g., see [5,26]), the violations or successful executions are represented as traces containing information of the composition activities. A set of such traces from previous executions may be used to derive new test cases for online testing. Furthermore, monitoring may be used to parametrize the test cases. As the configuration of tests may depend on the operational context of the application, such context information can be provided by monitoring.

## References

1. Baresi, L., Ghezzi, C., Guinea, S.: Towards Self-healing Service Compositions. In: First Conference on the PRIncipled of Software Engineering (PRISE 2004), pp. 11–20 (2004)
2. Baresi, L., Guinea, S., Pasquale, L.: Self-healing BPEL processes with Dynamo and the JBoss rule engine. In: ESSPE 2007: International workshop on Engineering of software services for pervasive environments, pp. 11–20 (2007)
3. Erradi, A., Maheshwari, P., Tosic, V.: Policy-Driven Middleware for Self-adaptation of Web Services Compositions. In: ACM/IFIP/USENIX 7th International Middleware Conference, pp. 62–80 (2006)
4. Modafferi, S., Mussi, E., Pernici, B.: SH-BPEL: a self-healing plug-in for Ws-BPEL engines. In: 1st workshop on Middleware for Service Oriented Computing, pp. 48–53 (2006)
5. Spanoudakis, G., Zisman, A., Kozlenkov, A.: A Service Discovery Framework for Service Centric Systems. In: SCC 2005: Proceedings of the 2005 IEEE International Conference on Services Computing, pp. 251–259 (2005)
6. Siljee, J., Bosloper, I., Nijhuis, J., Hammer, D.: DySOA: Making Service Systems Self-adaptive. In: Benatallah, B., Casati, F., Traverso, P. (eds.) ICSOC 2005. LNCS, vol. 3826, pp. 255–268. Springer, Heidelberg (2005)
7. van der Aalst, W.M.P., Pesic, M.: Specifying and Monitoring Service Flows: Making Web Services Process-Aware. In: Baresi, L., Di Nitto, E. (eds.) Test and Analysis of Web Services, pp. 11–55. Springer, Heidelberg (2007)
8. Günther, C.W., van der Aalst, W.M.P.: Fuzzy Mining - Adaptive Process Simplification Based on Multi-perspective Metrics. In: Business Process Management, 5th International Conference, BPM, pp. 328–343 (2007)
9. Nezhad, H.R.M., Saint-Paul, R., Benatallah, B., Casati, F.: Deriving Protocol Models from Imperfect Service Conversation Logs. IEEE Transactions on Knowledge and Data Engineering (TKDE) (to appear, 2008)
10. Wang, Q., Quan, L., Ying, F.: Online testing of Web-based applications. In: Proceedings of the 28th Annual International Computer Software and Applications Conference (COMPSAC), pp. 166–169 (2004)

11. Deussen, P., Din, G., Schieferdecker, I.: A TTCN-3 based online test and validation platform for Internet services. In: Proceedings of the 6th International Symposium on Autonomous Decentralized Systems (ISADS), pp. 177–184 (2003)
12. Chan, W., Cheung, S., Leung, K.: A metamorphic testing approach for online testing of service-oriented software applications. International Journal of Web Services Research 4, 61–81 (2007)
13. Bai, X., Chen, Y., Shao, Z.: Adaptive web services testing. In: 31st Annual International Computer Software and Applications Conference (COMPSAC), pp. 233–236 (2007)
14. Bai, X., Xu, D., Dai, G., Tsai, W., Chen, Y.: Dynamic reconfigurable testing of service-oriented architecture. In: Proceedings of the 31st Annual International Computer Software and Applications Conference (COMPSAC), pp. 368–375 (2007)
15. Canfora, G., di Penta, M.: SOA: Testing and Self-checking. In: Proceedings of International Workshop on Web Services - Modeling and Testing - WS-MaTE, pp. 3–12 (2006)
16. Ruth, M., Oh, S., Loup, A., Horton, B., Gallet, O., Mata, M., Tu, S.: Towards automatic regression test selection for web services. In: Proceedings of the 31st Annual International Computer Software and Applications Conference (COMPSAC), pp. 729–734 (2007)
17. Ruth, M., Tu, S.: A safe regression test selection technique for Web services. In: Second International Conference on Internet and Web Applications and Services (ICIW) (2007)
18. Di Penta, M., Bruno, M., Esposito, G., et al.: Web Services Regression Testing. In: Baresi, L., Di Nitto, E. (eds.) Test and Analysis of Web Services, pp. 205–234. Springer, Heidelberg (2007)
19. Martin, E., Basu, S., Xie, T.: Automated Testing and Response Analysis of Web Services. In: IEEE International Conference on Web Services (ICWS), pp. 647–654 (2007)
20. Bai, X., Dong, W., Tsai, W.T., Chen, Y.: WSDL-Based Automatic Test Case Generation for Web Services Testing. In: Proceedings of the IEEE International Workshop on Service-Oriented System Engineering (SOSE), pp. 215–220. IEEE Computer Society, Los Alamitos (2005)
21. Tarhini, A., Fouchal, H., Mansour, N.: A simple approach for testing Web service based applications. In: Bui, A., Bui, M., Böhme, T., Unger, H. (eds.) IICS 2005. LNCS, vol. 3908, pp. 134–146. Springer, Heidelberg (2006)
22. Lübke, D.: Unit Testing BPEL Compositions. In: Baresi, L., Di Nitto, E. (eds.) Test and Analysis of Web Services, pp. 149–171. Springer, Heidelberg (2007)
23. Dong, W.L., Yu, H., Zhang, Y.B.: Testing BPEL-based Web Service Composition Using High-level Petri Nets. In: EDOC 2006: Proceedings of the 10th IEEE International Enterprise Distributed Object Computing Conference, pp. 441–444. IEEE Computer Society, Los Alamitos (2006)
24. Pernici, B., Metzger, A. (eds.): Survey of quality related aspects relevant for SBAs. S-Cube project deliverable: PO-JRA-1.3.1 (2008), http://www.s-cube-network.eu/achievements-results/s-cube-deliverables
25. Suliman, D., Paech, B., Borner, L., Atkinson, C., Brenner, D., Merdes, M., Malaka, R.: The MORABIT approach to runtime component testing. In: Proceedings of the 30th Annual Int'l. Computer Software and Applications Conference (COMPSAC), pp. 171–176 (2006)
26. Barbon, F., Traverso, P., Pistore, M., Trainotti, M.: Run-Time Monitoring of Instances and Classes of Web Service Compositions. In: IEEE International Conference on Web Services (ICWS 2006), pp. 63–71 (2006)

# The inContext Pervasive Collaboration Services Architecture*

Stephan Reiff-Marganiec[1], Hong-Linh Truong[2], Giovanni Casella[3], Christoph Dorn[2], Schahram Dustdar[2], and Sarit Moretzky[4]

[1] Department of Computer Science, University of Leicester, UK
srm13@le.ac.uk
[2] Distributed Systems Group, Vienna University of Technology, Austria
{truong,dorn,dustdar}@infosys.tuwien.ac.at
[3] Softeco Sismat SpA, Italy
giovanni.casella@softeco.it
[4] Innovation Lab, Comverse, Israel
Sarit.Moretzky@comverse.com

**Abstract.** Traditional collaborative work environments are often proprietary systems. However, the demands of todays e-worker are such that they use their own tools and services and collaborate across company boundaries making highly integrated solutions less feasible. Service oriented computing provides an obvious solution here, in providing mechanisms to loosely integrate many tools and services. In this paper, we present the inContext PCSA (Pervasive Collaboration Services Architecture), which is a reference architecture for building context aware collaborative systems that are based on service oriented techniques.

## 1 Introduction

Collaborative systems are tools supporting collaborative work, typical examples are document management systems or customer information systems where different staff of the same organisation can access information and contribute to information in order to jointly bring forward the aim of the organistion. Many of the existing collaborative systems are not integrated with each other, so for example workflow and document management are not connected, or the communications systems are entirely separate from the other two. This means that information either needs to be transferred manually (e.g. logging call activities in a workflow system), or is simply not available where and when it is required. Clearly this calls for an infrastructure that allows for integration of the different activities. The other major disadvantage is that systems are usually used within a single organization, while nowadays collaborative work often spans institutional boundaries calling for platforms that operate across these boundaries. A further disadvantage of existing systems is that they are not context aware, that is the user's context is not automatically available to support the given activities.

* This work is supported by inContext (Interaction and Context Based Technologies for Collaborative Teams) project: IST IST-2006-034718.

© Springer-Verlag Berlin Heidelberg 2008

Work in the past has addressed some of the above aspects, in particular the issue about context. However, it has not done so by considering a single platform addressing all needs – which partly might have been due to limitations with the available middleware and underlying systems. In the inContext project we have developed a platform, or reference architecture called inContext PCSA (Pervasive Collaboration Services Architecture) which provides a context aware setting in which independent collaborative services, exposed as web services, can be used to build different collaboration tools. The platform provides a novel middleware layer that allows selection and composition of services at runtime to fulfil a users current needs. It is pervasive in both the sense that the user does not need to be aware of how and which services are selected, they simply achieve the desired aim by using systems build on top of the platform and thath services can run on different types of devices or platforms. To achieve this service selection, and in particular to identify the most suitable service for a user in their current context the platform is context aware. Context here covers not only location and devices, but also activities creating the all-essential link for integrating collaborative artifacts and information.

**Overview.** This paper is structured as follows: the next section provides a motivating example and background information on context aware and collaborative systems. We then turn our attention to the PSCA by providing an overview and discussing the essential subsystems. Before drawing concluding remarks and discussing further work, we present examples of the use of the architecture.

## 2 Motivation and Background

In terms of motivation, let us consider a scenario that is typical in the area of collaborative work, and which calls for many of the PCSA features presented in this paper. Imagine a user in an internal project $A$ in his company requires a document relevant for his current activity. The PCSA will utilize document search service $dsA$, through which the documents for the project can be accessed. Later the same user switches to project $B$ – a joint venture between his own company and an external association. Again the user request a document search service, this time the platform will use $dsB$, a public repository service with access by subscribers. We can see that in this simple example based on the user context (mostly his activity in this case) differing services are selected as most appropriate and are consequently invoked in a transparent manner to achieve the user's aim. Behind the scenes there is much technical activity and usual examples are much larger: the document search might be part of a meeting planning tool or a governmental policy review systems.

A vast number of tools supporting collaborative work is available today, as indicated in [1,2,3]. However, many of these tools are used within a single organization's boundaries, while many emerging collaborative work scenarios are spanning these boundaries. When we consider the dynamic and distributed collaboration spanning different organizations and the support of user participation/customization required we find that existing collaborative tools are simply insufficient. There is a number of requirements that emerging collaborative tools need to support. First, there is a need to utilize different collaboration tools which can be provided and hosted by different organizations. Second, there is a need to adapt collaboration tools to the context of the collaboration as we have seen in the motivating example earlier on. Existing collaboration tools provide rich features which work very well when utilised in isolation, for example, for sharing documents or for managing activities. However, those features are not easily integrated into a single collaboration toolset, where various features are required to accomplish the collaboration for various reasons, such as they are propriety collaboration tools, they provide no open interface, or they are tightly-coupled systems.

Dynamic user-defined and user-customized collaboration calls for well-defined, common collaboration services which can be easily composed and adapted for different collaboration contexts. We have found that commodity of CWE services is in increasing use and open standards are widely employed for collaboration tools [3]. However, there are many open questions that need addressing: How can we support diverse collaboration services that can be composed and customized? How can we enable context-awareness and collaboration adaptation across organizational boundaries? How can we support the user to perform collaboration from anywhere with any device? These questions motivated us to investigate the concept of commodity collaboration services which are composable and common. We approach this concept by utilizing Web services technologies and activity-based collaboration techniques to develop a so-called pervasive collaboration services architecture (PCSA).

## 3 Overview of the PCSA

The inContext Architecture, also referred to as PCSA (Pervasive Collaboration Services Architecture), consists of three main parts: the user applications, the collaboration services and the inContext core platform [4] - we consider these in turn. To provide an overview, the overall architecture is depicted in Figure 1.

User applications have not been at the forefront of the development here, but our case studies have been supported with respective frontends. Applications at this level are meant to be used by the collaborative worker, so some effort has been spent on considering user applications technologies for both mobile and stationary workers.

Underpinning the environemnt are collaborative services: essentially any service (implemented as a web service) or any software whose interfaces have been exposed as web services that have a meaning for collaborative work. Services at this level are, for example, a Document Service (allowing to retrieve and collate documents that are relevant to the current user activity), an Activity Service (concerned with any information regarding activities), or an User and Team Management Service (providing all sorts of team related information).

The core platform contains four main ingredients: the Access Layer, the Context Management, the Service Management and the Interaction Mining. Each of these is developed as services, providing much flexibility to the platform.

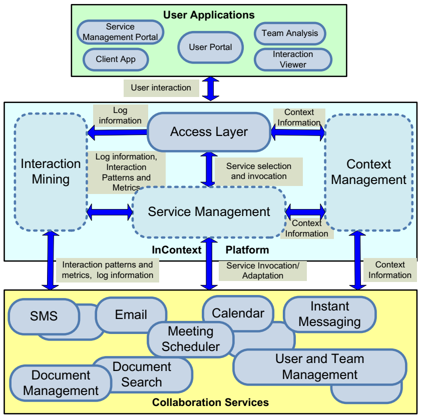


**Fig. 1.** An overview of the inContext system [4]

The Access Layer controls access to the platform by checking user credentials – it is the main entry point to using the inContext platform. However, its functionality goes beyond authentication: any transaction across the Access Layer is tracked to gather information on user activities (which in turn enrich the context information) and user decisions (e.g. when a number of services was available it is monitored which was actually selected by the user).

The Context Management sub-system is concerned with gathering, storing, providing and enriching context information, but also provides mechanisms to retrieve context from the store. Furthermore, reasoning techniques allow deriving new, richer context information from the information available in the store. The contex model implementation is based upon RDF triples and the reasoning is based on an enhanced version of the Jena engine. Interaction Mining provides additional information by considering past behavior. Context management and interaction mining are beyond the scope of this paper[1].

The Service Management constitutes the part of the inContext platform most relevant for this paper, as it is here were the collaboration services are drawn together and composed into rich context aware collaboration tools on demand. Service Management is concerned with registration and lookup of services and their execution – this goes beyond standard web service functionality by identifying the best suitable choice for the users current context and activity based on non-functional attributes of services.

## 4 Common SOA-Based Collaboration Services

The inContext PCSA is based on SOA to allow for collaboration services to be dynamically located and invoked. The platform supports flexible and dynamic

[1] inContext D2.2 and D2.3 discuss these in detail; both are available at www.in-context.eu.

**Table 1.** Examples of common collaboration services

| **Collaboration Services** | **Description** |
|---|---|
| `User and Team Management Service` | provides a list of projects related to a user with detailed information on project members, timeline and team structure. |
| `Member Search Service` | searches for relevant people based on specific role or expertise. |
| `Personal Document Search Service` | allows searching for text patterns inside a set of documents stored on specific hosts which were declared as shared for the inContext platform. |
| `Document Service` | manages virtual 'shared areas' for documents. |
| `Short Message Service` | enables to send SMS to mobile users. |
| `Meeting Scheduling Service` | is a composed service allowing setting up a meeting. |
| `Activity Service` | manages the activities associated to a project, enabling the creation and the organization of such activities in an activity tree and assigning them to users, documents, locations, etc. |

collaborative working environments able to aggregate heterogeneous services while at the same time considering and exploiting the user's context.

Common collaboration services can be developed and provided by different organizations, following well-defined interfaces to support fundamental tasks typically required by collaboration tools. Based on the inContext's case studies, we have analyzed requirements for collaboration platforms and identified various common collaboration services and have as part of the platform provided a set of such services. Although, these services were demanded by the inContext case studies, they are not specific to these. The services are more general and can be used in a wide variety of emerging collaborative work scenarios as well as be served as basic services for building different collaboration tools. Table 1 presents a selection to show the flavour of typical collaboration services.

Common collaboration services can be atomic or composite – as is typical for web services. By utilizing well-developed publish-registry techniques in Web services, common collaboration service will be registered in a repository named Service Registry. However, when registering a service some extra information is required: All services (actually each operation) are organized in categories, within the Service Registry. Each registered service belongs to at least one category. The category is important, as it is this which is used for lookup; each category also has associated non-functional attributes and a well defined generic service interface. Then, common collaboration services can be used to build collaboration tools which could discover and execute the services based on collaboration context.

## 5 Context-Aware Service Management

The context-aware service management identifies the most appropriate services based on the user's context and current needs and composes these into a collaborative tool-set. There are three aspects here that go beyond what standard service oriented techniques offer: (1) we have addressed the need to select the most appropriate service automatically, (2) we have addressed the requirements for selecting services as part of a worklfow and (3) we have provided techniques to invoke services through a standard interface by automatically mapping specific service interfaces to more generic interfaces that are exposed to the user application. We will consider these 3 aspects in the next few sections.

*Selecting Services.* One of the challenging aspects in service oriented computing is selecting the best service if a number of services are on offer. We devised a ranking mechanism called the RelevanceEngine which ties in with service lookup in the service management subsystem. The RelevanceEngine has one job: to consider a list of suitable services that all seem to functionally address the user's requirements and rank these so that the user can see which service is most appropriate for supporting their activity in their current situation.

The ranking mechanism makes use of a number of inputs: it queries the context management system to obtain information about the user's context; it queries the data mining component to gain an insight into historical handling of a similar situation and it of course explores the services profile – the extra meta data in the registry that is associated with the service category. For example, if the user is looking for a printing service, the meta data will tell us static information such as whether the service is colour or not and how fast the printer is; it will also provide a query URL to find out about the current service use (e.g. queue length). The user's context will amongst others provide insight into whether he requires the printout very quickly and whether the document is very long. All these facts are combined and a rank value is calculated for each service by using an automated, type based version of LSP (Logic Scoring for Preferences). This paper provides a wider overview of the architecture and a very detailed discussion of the ranking mechanism is beyond the scope, however this has extensively been described in [5]. Briefly, the LSP mechanism can handle large numbers of criteria while maintaining an assurance that even factors with a small weight but which a user cares strongly about are given appropriate weighting in the resulting score. It can also act as both "ranker" and "filter" – that is we consider hard and soft criteria using the same mechanism and ensure that services which do not meet hard criteria are shown as inappropriate in the ranking (essentially a score of 0): for example a hard criteria might be "the service must be free", while the related soft criteria would be "the service should cost as little as possible".

It is worthwhile pointing out that we are using a pragmatic extension to service models in a standard UDDI repository: we have developed a model for capturing key non-functional data about services in this way. This mechanism is lightweight and requires little technologythat goes beyond standard web services; in particular it only requires for a service developer to register a few extra values in the repository. Of course one could consider semantic web technologies here, which provide mechanisms to express properties but these are a more fundamental shiftin the technology used and hence we decided against them. Independent of this, the data about services is an input to the ranking mechanisms and the same would be still appropriate in the context of semantic web services with richer service descriptions.

*The Composition Context.* The mechanism just described was initially developed to find the most appropriate service for a given activity, considering the user's context but not necessarily the context of execution of the service. However, usually services are not required in isolation but are often invoked as part of a workflow. The ranking mechanism has been extended by what is called composition context, essentially information gathered about the last stages in the workflow that we have executed: did services from a certain provider fail? Are there policies that disallow us to select (or would mean preferential treatment if we chose) a future service from a specific provider? The concept and related ranking mechanism have been described in [6], but the idea is probably best explained with a small example: consider ordering a book. You have a choice of two providers, provider A charges €10 for the book, provider B €13. If we select the most optimal single service, we would select provider A. But in our workflow context, we know that the book also has to be shipped to us and find that provider A charges €5 for shipping, while provider B offers it for free, thus overall provider B is the better choice. This example is simple but it is only meant to show that local optima differ from global ones; further one could argue that some websites currently already provide such functionality: they usually do so by considering services offered by the same provider. The composition context explored here is not bound to just relating information by the same provider, but can be applied to services from different providers.

*Mapping Interfaces.* Considering that we retrieve services from all sorts of providers at runtime, we must ensure that they can be invoked by the platform in a coherent way. In order to achieve this, we have assigned to each category a common interface and each service in that category has to provide mapping information on how to map the service interface to the common interface. This way we enable transparent execution of various services with different interfaces for a specific category.

To support this mechanism of service interface mapping, we implemented a service called Interface Mediator. This service forwards web service calls after applying a transformation in order to match the destination web service interface. As already mentioned, if a service is registered under a certain category the consumer should use the common interface to use service of this category. During runtime the interface mediator relies on XSLT style sheets to map the common interface to a service interface. This allows the service providers to perform some complex data manipulation to match their requirements. In order to expose the PCSA capabilities, we created some XSLT templates that can be directly used to (1) query and use context data, (2) gather user preferences / information or (3) Query any service metadata.

While offering high flexibility and enabling scenarios like translating content based on user's language, sending an SMS to the relevant phone number, or converting data to the right format (e.g. currency units, or temperature scales), this transformation implies of course an overhead compared to a direct call. Its impact on the performance is very limited for several reasons.

– The style sheet is compiled to machine code allowing for very fast execution.
– The data is manipulated at the XML level without being marshaled to any programming model which removes expensive conversions.
– The external service calls go through the Access Layer anyway in order to be logged and to provide feedback to the system. Integrating the interface mediator there implies no network overhead for the transformation itself.
– XSLT is standard technology with very fast engines emerging.

The overheads are outweighed by the benefits of being able to dynamically select a service based on the current situation. This mechanism makes the platform more robust (being able to use another service if one fails), but also increases its adaptability by offering to add/ remove services to existing categories.

## 6 Context-Aware Collaboration Services

In typical service composition one considers functional and QoS parameters. Collaborative work scenarios require in addition for context to be considered as a main criteria. In this section we address two issues: context-aware composition and adaptation support for collaboration services.

*Context-aware Composition.* The composition of collaboration services is based on collaboration context. By utilizing collaboration services, collaboration tools can be built. In our PCSA, a collaboration is described by collaboration activities which are performed by a set of users. Consequently, a collaboration context will be determined when the activities are specified and the context will be updated by the user or by the services which monitor actions within activities.

A tool supporting the end user to collaborate can utilize collaboration services, thus it has to manage compositions for collaborations. During the collaboration, the user will specify activities which include information about who is involved in them, which artifacts are needed, the type of collaboration services used, etc. Specified activities are managed by the Activity Service. All context information related to activities are managed, for example, the location of involved people and the status of services being used for the activities. When collaboration services are required, the most appropriate services will be determined and composed based on the current collaboration context; we discussed the technical mechanism for service ranking in the previous section.

*Supporting Adaptation based on Collaboration Context.* Collaboration services are deployed as web services. These services must be aware of changes in the collaboration context, and therefore they need access to the current collaboration context. The PCSA supports two types of context-based adaptation: (1) service-level adaptation focuses on improving the behavior of the invoked service depending on provided context information and (2) composition-level adaptation aims at selecting and combining the most suitable set of services to fulfil the user's requirements in the given situation.

To this end, we provide a generic solution for distributed collaboration context sharing. The sharing mechanism remains context model agnostic. While the actual context information is managed by the context management framework, the context sharing mechanism is responsible for providing correlation information. This correlation information acts as the initial context entry point, thereby allowing a service to retrieve the relevant information from the context store. To remain independent of service interface and Web service stack, we insert the correlation information in the header part of a SOAP message. Whenever a service operation is invoked, the message header includes the URI of the invoking user and the user's current activity. This provides sufficient correlation for a service to obtain the context information and adapt its operation, if needed.

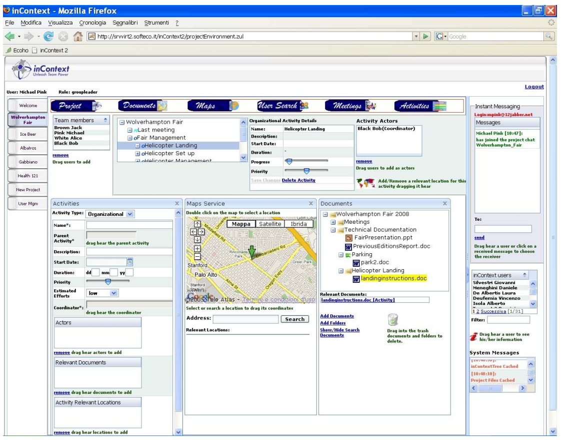
One of the main advantages of using Web services technologies for common collaboration services is the ability to loosely couple context information: (1) services do not need to explicitly pass along large sets of context information of which each individual service requires potentially only a small subset; (2) services need to understand only that part of the overall context model that they require; and (3) extensions for domain specific collaboration services (e.g., health care) can place additional correlation information in the SOAP header without having to update existing services.

Utilizing the SOAP header extension for context correlation yields another benefit to simplify cross-organizational collaboration. SOAP intermediaries such as the Access Layer but also message routers, security checkpoints, and governance mechanism for SLAs in general can access the header information and subsequently base their decision on the available context instead of inflexible policies and rules. Thus, adaptation at the service composition level becomes increasingly feasible and manageable. For example, consider the following adaptation supported by the PCSA: (a) The Access Layer forwards a notification request to the most suitable communication service based on user preferences, costs, and available devices. (b) The service interface mediator is able to extract missing data from the context to invoke a service demanding for additional input parameters not specified by the generic service interface. (c) Once a shared document space is selected for an activity, all subsequent service calls are forwarded to the same service endpoint.

## 7 Experimental Evaluation

In this section, we discuss how we utilize PCSA to build different collaboration tools. We present our experiences based on the development of two real case studies proposed by our end-user partners:

**Scheduling a meeting:** The Electrolux Group is one of the world's largest manufacturers of white goods. Within the company secretaries must often organize meetings which is a difficult task. In fact, managers are often unavailable to participate to meetings due to the multiple activities in which they are involved, in which case the secretary can select his/her project proxy which requires an understanding of the project team structure. Moreover it



User: Michael Pink Role: groupleader

Team members: Brown Jack, Pink Michael, White Alice, Black Bob

Drag users to add

Wolverhampton Fair
- Last meeting
- Fair Management
  - Helicopter Landing
  - Helicopter Set up
  - Helicopter Management

Organizational Activity Details

| | |
|---|---|
| Name: | Helicopter Landing |
| Description: | |
| Start Date: | |
| Duration: | - |
| Progress | |
| Priority | |

Activity Actors: Black Bob(Coordinator)

Drag users to add as actors

Add/Remove a relevant location for this activity dragging it hear

Login:mpink@12jabber.net

Michael Pink [10:47]: has joined the project chat Wolverhampton_Fair

Drag hear a user or click on a received message to choose the receiver

Activity Type: Organizational

Name*:

Parent Activity* drag hear the parent activity

Description:

Start Date:

Duration: dd mm yy

Priority

Estimated Efforts low

Coordinator*: drag hear the coordinator

Actors — remove drag hear actors to add

Relevant Documents — remove drag hear documents to add

Activity Relevant Locations — remove drag hear locations to add

Double click on the map to select a location

Select or search a location to drag its coordinates

Address:

Relevant Locations:

Wolverhampton Fair 2008
- Meetings
- Technical Documentation
  - FairPresentation.ppt
  - PreviousEditionsReport.doc
  - Parking
    - park2.doc
  - Helicopter Landing
    - landinginstructions.doc

Relevant Documents: landinginstructions.doc [Activity]

Drag into the trash documents and folders to delete.

inContext users: Silvestri Giovanni, Meneghini Daniele, De Albertis Laura, Deufemia Vincenzo, Isola Alberto

1 2 Successiva [1/31]

Drag hear a user to see his/her information

inContextTree Cached
[10:48:10]: Project Files Cached

**Fig. 2.** The Event Management Tool

is often difficult to communicate with them (e.g. to establish a meeting date) due to their travels (they are frequently in different time zones, are not always able to access the web or to answer phone calls).

**Wolverhampton Fair:** Every year the West Midlands Local Government Association organizes and manages the Wolverhampton fair. In particular a manager must build the staff to manage the fair (the workers belong to different departments and have different skills and experiences), must assign the activities related to the fair and must check that during the fair coordination and communicating between all staff is as expected, which requires tools to exchange documents and to communicate quickly.

The two usage scenarios are addressing very specialised needs which are very different in their requirements. We were able to exploit the PCSA successfully to build two user applications to handle these scenarios. The two user tools exploit a subset of common collaborative services, which are composed and adapted in different ways to satisfy the user needs. Figure 2 shows the GUI of the Event Management Tool as an example. The Event Management Tool offers a set of tools to organize events but is also applicable to manage general projects. The Meeting Scheduling Tool addresses the needs of the Electrolux case study.

The following shows some collaboration services used for each of the two tools, where SM and EM are used for the *Schedule a Meeting Tool* and the *Event Management Tool* respectively; the last three servcies are composite.

**User & Team Management Service** is used to retrieve participants details and project team structures (SM) and to maintain the event stuff structure and to retrieve user details, skills, experiences, etc. (EM)

**Activity Service** is used to explore the project activity tree in which the meeting is created (SM) and to create the activities that must be handled in the event and to assign such activities to the staff members (EM).

**Document Service** is used to retrieve information about relevant documents for a meeting and to store the meeting agenda (SM) and to share and organize documents (EM).

**The Relevant Documents Finder** is a composition of the activity service and the document service. Based on relations between the current user activity (e.g. the meeting that is scheduled and a general activity) and others project activities it retrieves the documents associated to these activities. Some reasoning rules are applied to identify the relevant documents.

**The Relevant User Finder** is a composition of of the activity service and the User & Team management service. Based on the relations between the current user activity and others project activities it provides information about users involved in these activities. Again, some reasoning rules help to ensure that all appropriate users are identified.

**The Notification Service** is a composition of the Instant Messaging Service, the SMS and the Mail Service. By exploiting the user context this service decides which is the best way to notify a user about something (e.g. an important message informs that he must attend a meeting) and sends the message using the best service selected.

Collabration services become sometimes unavailable (this is unavoidable in a distributed, dezentralised environment). In these cases the PCSA lookup mechanism enables selection of an alternative service in a manner transparent to the user by using context and the interface mediator.

Our experimental evaluation shows that the PCSA enables the exploitation of a set of collaboration services to build heterogeneous collaboration tools addressing different requirements and functionalities. Moreover, by using the service common interfaces composition of simple services to offer new complex services which are able to satisfy the user needs is possible. Finally the dynamic management of services enables integration of different services with the same functionalities and to exploit them according to their availability.

While we considered two specific cases here – both taken from the domain of collaborative work – these are only meant to illustrate the ideas. The platform developed addresses the need of collaborative systems which we have analysed and briefly introduced at the start of the paper and hence allows for the dynamic building of tools for collaborative work environments which tend to require flexible system structures where the system takes much of the burden of providing the right service at the right time based on the users activity. Many of the developed mechanisms can be applied outside collaborative systems, for example the approach for service selection has not been developed with only collaboration services in mind, but rather with a wider view of service slection.

## 8 Related Work

Many basic collaboration tools and services, such as document sharing, calendars, and instant messaging have been developed. However, currently it is not easy to compose these services and make them interoperable for Web-based, user-customized collaborations because most of them lack well-defined Web services interfaces. Web services support have been incorporated into few collaboration services such as BCWS, Google Doc, and Microsoft Sharepoint for document sharing. In our work, we not only propose solutions for the interoperability of collaboration services but also present how common collaboration services can be composed suitable for different collaborations based on context.

Recently, various researchers have advocated the standardization of (basic) collaboration services. A CoCoS Working Draft[2] proposes common collaboration services in terms of message representation, service operations and service behaviors. CoCoS addresses a subset of common collaboration services proposed in our PCSA but does not discuss the composition and execution aspects of collaboration services. The ECOSPACE project[3] also investigates various common collaboration services. However, it focuses on document sharing services. The OCA-WG (Open Collaborative Architecture Working Group[4]) aims at defining a reference architecture for collaboration services.

In the area of service selection [7] propose the addition of a broker component to the service selection architecture which essentially sits between the UDDI repository and the invoker and monitors service invocations and the resulting quality. However they, like most other approaches (e.g. [8]), only consider service quality as criteria for service ranking. We have signifcantly added to this with the more complex context model that is used in our work. Also, we have extended the Data model in the repository itself to get richer service information.

## 9 Conclusion and Further Work

The lack of common collaboration services and an open architecture for collaborative working environments hinders the integration and reusability of diverse collaboration tools. We have presented the inContext PCSA – a reference architecture for context aware collaborative systems that defines and provides an open platform with various common collaboration services. The PCSA allows to combine collaboration services into larger platforms that fulfil the needs of an organisation or user. What should be noted is that the combination is loose, in the sense that the actual service for a specific task is only selected at runtime based on the users activites and current needs.

What constitutes a collaboration service is open: in principle any service could be used as part of a collaboration and can be easily introduced to the platform by the creator of the service registering the same and providing some meta data. The PCSA itself is also built from services and hence can be used in its entirety, or selected components can be used within other contexts.

[2] http://www.ubicollab.net/images/stories/UbiCollab/Standards/CoreCollaboration Services_v0.1.pdf

[3] http://www.ip-ecospace. org

[4] http://www.oca-wg.org

The resulting technical platform has been used to implement two quite diverse toolsets for two real case studies and initial test results have been discussed. Currently we are running extensive end-user tests by exposing the toolsets to larger audiences. Another line of future work is improvements at the level of individual system components, such as further refinement of the ranking mechanisms or enhancement of service profiles.

## References

1. O'Leary, D.E.: Wikis: From each according to his knowledge. Computer 41(2), 34–41 (2008)
2. Optaros, Inc.: Unleashing the power of open source in document management. Optaros Whitepaper (2006), `http://www.optaros.com/system/files/optaros_wp_os_crm_20060316%282%29.pdf`
3. Skopik, F., Truong, H.L., Dustdar, S.: Current and Future Technologies for Collaborative Working Environments, ESA Report (2008), `https://www.vitalab.tuwien.ac.at/autocompwiki/index.php/Main_Page`
4. Truong, H.L., Dustdar, S., Baggio, D., Corlosquet, S., Dorn, C., Giuliani, G., Gombotz, R., Hong, Y., Kendal, P., Melchiorre, C., Moretzky, S., Peray, S., Polleres, A., Reiff-Marganiec, S., Schall, D., Stringa, S., Tilly, M., Yu, H.: Incontext: A pervasive and collaborative working environment for emerging team forms. In: SAINT, pp. 118–125. IEEE Computer Society, Los Alamitos (2008)
5. Reiff-Marganiec, S., Yu, H.Q., Tilly, M.: Service selection based on non-functional properties. In: NFPSLASOC 2007. LNCS. Springer, Heidelberg (in press, 2007)
6. Yu, H., Reiff-Marganiec, S., Tilly, M.: Composition context for web services selection. In: ICWS 2008 (in press, 2008)
7. Al-Masri, E., Mahmoud, Q.: Discovering the best web service. In: Proceedings of the 16th international conference on World Wide Web, pp. 1257–1258. ACM, New York (2007)
8. Seo, Y., Jeong, H., Song, Y.: A study on web services selection method based on the negotiation through quality broker: A maut-based approach. In: Proceedings of International Conference on Embedded Software and Systems, pp. 65–73 (2004)

# Leveraging the Upcoming Internet of Services through an Open User-Service Front-End Framework

David Lizcano[1], Miguel Jiménez[1], Javier Soriano[1], José M. Cantera[2], Marcos Reyes[2], Juan J. Hierro[2], Francisco Garijo[2], and Nikolaos Tsouroulas[2]

[1] Universidad Politécnica de Madrid
28660 - Boadilla del Monte, Madrid, Spain
{dlizcano,mjimenez,jsoriano}@fi.upm.es
[2] Telefónica I+D
28043 - Emilio Vargas 6, Madrid, Spain
{jmcf,mru,jhierro,fgarijo,nik}@tid.es

**Abstract.** The Internet of the Future is expected to be composed of a mesh of interoperable Web Services accessed from all over the Web. This approach has not yet caught on since a global user-service interaction is still an open issue. This paper states our position with regard to the next generation front-end technology for the Internet of the Future. This approach will enable the massive deployment of services over the Internet in a user-centric fashion. This paper advocates the full development of front-end technologies to bring services closer to users, empowering them anytime and anywhere. It also outlines all the main gaps and technological challenges that have to be addressed. Finally, a model and an architecture are proposed for building these technologies into NESSI's Open Framework Reference Architecture, NEXOF-RA.

## 1 Introduction

One of Internet's future ambitions is the massive deployment of services and service-oriented systems across the whole Web. Actually, Web Services-based Service Oriented Architectures (SOAs) have gained momentum over the last few years [1]. And SOAs are expected to be the key to the user-service interaction take-off. However, these solutions are currently confined to company boundaries, and the desired global provision and consumption of user-centric compositional services from the envisioned Internet of Services is still at an early stage [2]. Despite high expectations, the emergence of a mesh of interoperable Web Services that could foster the massive deployment of user-friendly services is continuously shattered by a series of well-known shortcomings [3], such as high technical complexity, implementation and maintenance costs, inflexibility and lack of widely accepted standards and open service frameworks. These issues are understandable considering that the underlying technologies were spawned by a machine-to-machine approach, not featuring a "face" for human users[4].

© Springer-Verlag Berlin Heidelberg 2008

This paper states our position, as Chairs and members of the NESSI User-Service Interaction Working Group (NESSI-USIWG), with regard to the next generation front-end technology for the Internet of the Future. This technology will enable the massive deployment of services over the Internet in a user-centric fashion [5]. In this paper we advocate the full development of front-end technologies to bring the services closer to users, empowering them anytime and anywhere. We outline all the main gaps and what technological challenges have to be addressed. We express in the paper what the NESSI USIWG is about to deliver as a research alignment in a forthcoming Position Paper, and we propose both a model and an architecture for building this technology. The outcomes and the vision presented in this deliverable will become part of NESSI's Open Framework Reference Architecture (NEXOF-RA): a coherent and consistent open service framework leveraging research in the area of service-based systems to consolidate and trigger innovation in service-oriented economies. The overall goal of NEXOF-RA, and thus the guiding principle of this paper, is to deliver a coherent set of globally applicable technologies. These technologies are intended to provide Europe with a digital services infrastructure to improve service flexibility, interoperability and quality. In addition, NEXOF-RA will try to establish strategies and policies to speed up the dynamics of the services ecosystem, as well as to foster the safety, security and wellbeing of citizens by means of new societal applications.

The remainder of this paper is organized as follows. First we present the shortcomings of the current SOA technologies with a view to the desired Internet of Services. We then go on to illustrate the guiding principles to achieve our aim. A reference model and its architecture are also proposed. Finally, we explain other related work, the main conclusions that can be drawn from the ideas in this paper, and what we consider to be the work that needs to be undertaken in the future.

## 2 Current Shortcomings on the Road towards an Internet of Services

The massive deployment of user-centric services over the Internet demands services that must be accessible for all users (not only enterprise stakeholders). Therefore, services should flexibly and dynamically support common daily processes (both business processes carried out by companies and processes conducted by individuals or groups in their daily life) at any time[4]. Users will see the tools supporting their daily work replaced by composite applications based on Web Services, but traditional Web Services are not well enough tailored to users and their daily processes.Obviously, SOA, as it was originally conceived, represents an architecture focused fundamentally on a B2B context. It is weak for B2C problems, since it does not offer the best prospects for dealing with user-service interaction [3]. We can tackle its shortcomings from three different perspectives:

1. SOA's aim: Conventional SOAs merely aim to facilitate seamless machine-machine collaboration. SOA deployments are very abstract and invisible to users. Its customers of choice are medium-sized or larger corporations instead of normal end users along the long tail of Internet. Therefore, with SOA, normal Internet users with little IT expertise have not been able to easily retrieve and use services because services mostly reside within company boundaries and are only accessed for professional use in a corporate context.
2. SOA's technology: Apart from SOA's aims, this architecture relies on a set of complex standards that are not user friendly [1]. Because, technically speaking, SOA is extremely complex, there needs to be one or more expert players within the value chain to build and provide solutions for their customers. In contrast to this one-to-many value chain model of numerous SOA use cases (where one expert serves many clients), new value chains should begin to be mostly loosely coupled (many-to-many) networks of self-managed self-sufficient users who can offer and consume resources via the Web.
3. SOA's governance: Finally, SOAs are subject to clearly defined regulatory frameworks since they mostly exist in the corporate context. The design, provision, maintenance, and coupling of services must be compliant with legal frameworks. Therefore, they do not allow for the flexibility that the described new user-services interaction model appears to need.

## 3 Design Principles Enabling the Internet of Services

The evolution of Web 2.0 sites and applications is a testimony to the progress achieved to improve user relevance and service usability. However, current service front-ends are far from meeting end-user expectations [6]. Applications are still based on monolithic, inflexible, and unfriendly UIs. This is a serious obstacle for achieving the benefits of the Internet of Services. In order to build the next generation service front-end for this ecosystem of services we propose three guiding principles. These principles are further detailed in the following.

### 3.1 Enabling Users to Design and Share Their Operating Workspace and Applications

Users should be able to design and implement their own interfaces in a flexible and friendly manner. New generation front-ends should provide new tools aiding end-user UI creation and self-adaptation, while supporting a dynamic computing infrastructure [7]. Community-based collaboration tools should also be supported to satisfy the demands for secure social interaction and improve knowledge and resource sharing. More to the point, the following aspects should be covered [5]:

1. Empower users to select resources of their interest, annotate, and configure their own personalized operating environment

2. Promote user pro-activeness for creating new resources, to improve versions of resources, and share further expertise about these resources, their use, and their interrelationships
3. Provide social facilities to share services, results, knowledge and resources with other users
4. Foster social interaction by providing visual mechanisms to manage user communities, user identity, privacy and security constraints, and the information to be shared, annotated, or send to specific groups and individuals

The new generation of user-service front-ends should be based on helping users with the flexible composition of applications. Application and service front-ends will no longer be conceived as monolithic blocks, but as a set of interoperable service front-end components –called gadgets–, which are available from a catalogue. A catalogue of components or gadgets will be available for creating application or service front-ends by using and combining these building blocks to construct new components. Users should be able to create their own applications by combining different catalogue components without any help from IT experts or a thorough knowledge of the underlying infrastructure [8].

## 3.2 Businesses Need to Adapt to the New Reality

Today's competitiveness-driven business markets and the severe time-to-market restrictions on applications, specifically for enterprise IT systems, have increased the business needs to evolve applications to suit this new reality. They can be evolved through the following key guidelines:

1. Businesses need to embrace the Software as a Service (SaaS) model as an effective software-delivery mechanism [9]. This approach helps to reach a marketplace of services that can be composed to create unanticipated business solutions adapted to real needs.
2. Next generation Web Services ecosystems must respond to unforeseen business requirements that emerge or evolve spontaneously. This should be supported by new software development methodologies that ease the integration, adaptation and evolution of service-based applications.
3. Company boundaries must be eroded, evolving towards the Internet of Services vision. This approach can be split into two perspectives:
   (a) Next generation business systems should adopt a user-centric approach to take into account users. Users of these services are no longer just the company's employees; clients should also access the same business resources. This could even affect the entrepreneurial service-based workflows [10].
   (b) Collaboration between companies, irrespective of their size, must be fostered, thus increasing productivity and accelerating innovation [7]. The creation of collaborative services by integrating components from disaggregated companies will afford new business opportunities and improve the global service provided to end-users.

### 3.3 Context-Adapted User-Service Interaction

The proliferation of multiple Internet-enabled devices allows end users to access the Web anytime and anywhere. As a result, there is a wide range of situations in which a user might need to access these services, and they must therefore be provided the right user interface for the right situation. These situations can be defined as the context [11] in which access occurs.

Next generation service front-ends should take into account the following context aspects:

1. The delivery context, that is, a set of attributes that characterizes the environment in which a service is going to be delivered. Delivery context is a crucial aspect with regard to service front-ends, as it provides a clear indication of what the capabilities of the target device, web browser and network are. Such aspects play an important role in the end user experience. The information about these capabilities should be exploited by service front-ends to provide a harmonized experience adapted to the peculiarities of every delivery context.
2. Users and their circumstances: This aspect includes properties such as user identity, profile and roles, social network, tastes and personal preferences.
3. The surrounding environment, including the spatial location, speed, light conditions, temperature, level of noise, nearby objects/things...
4. The situation / time which has to do with variables such as date and time, weather, season, at home or at work, on vacation, on a business trip...

## 4 High Level Architecture

This section describes our overall proposed architecture for service front-ends (see Fig. 1). The architecture is consistent with the guiding principles and model. The central piece of the architecture is the NG-SOA Broker. The broker stores, indexes and references every available resource [12], empowering users by giving them access to context-aware functionalities, such as resource discovery and recommendation, knowledge sharing through a social network, establishment of marketplace relationships between resource providers and other user roles. Three high-level modelling phases are proposed:

– Delivery Access Layer
– Access Layer Composition
– Access Layer Development.

Each phase takes into account one of the above aspects. It defines models and standards in order to take into account the associated elementary building blocks (workspaces, gadgets, application services, content delivery services...) and all the possible interactions among them. This overview should also define the different user roles we envision in the NG-user/services interaction. These roles should specify how users will exploit the services, how they will adapt services

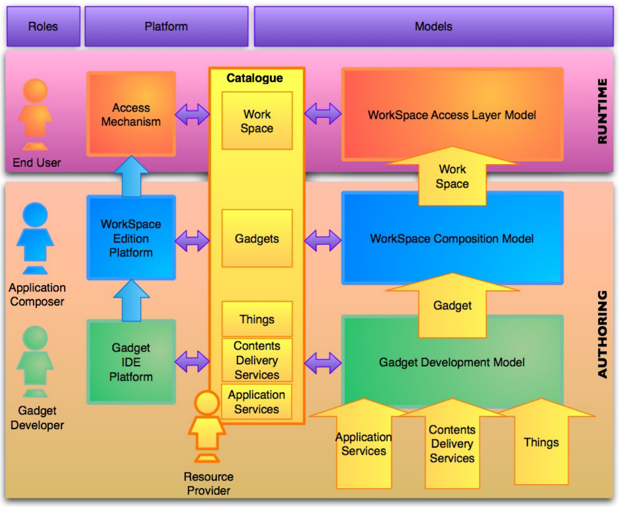


**Fig. 1.** High level architecture for the next generation of service front-ends

to their requirements and needs and how (and who) will develop the parts enabling this interaction. To address this functionality several tools must be defined to create, compose and exploit the different models. These tools are the Gadget Development Platform, Mashup Platform and the Access Mechanism.

In this scenario authors or developers can build their own service front-end access point or gadget by combining a universe of accessible resources. Users are not supposed to be technological experts, they will be domain experts able to define and solve concrete problems by combining the provided services and tools. To do so, users should employ the facilities offered by the Gadget Development Platform.

The Application Composer role will build the Service Front-End Layer by composing several gadgets to build a fully interoperable application. The application will be able to enact end user processes in a fast and dynamic manner and deal with situational applications.

It will be the end user that exploits the results of the developed applications. These users will not notice any difference between the conventional applications and the applications built according to the above development method. They will access applications suited to their context in a location-, device-, technology-independent manner.

## 4.1 Delivery Access Layer

The Delivery Access Layer will be in charge of adapting and enriching the service front-end to meet the restrictions imposed by a given context. Therefore, this layer is responsible for providing a harmonized user experience in every context, hiding back-end services and developers from the underlying complexities. The functionalities implemented by this layer include, but are not limited to:

– Delivery Context Detection using some kind of evidence such as the HTTP headers (User Agent, Accept, UAProf ...)
– Simple Content and Application Adaptation depending on the Delivery Context: sending the correct markup, style sheets and script code, selecting the most appropriate resources taking into account restrictions such as the supported image formats, or the dimensions of the screen, and so on.
– Advanced (Semantic) Content and Application Adaptation. This functionality has to do with exploiting the context and application semantics (knowledge) to perform advanced adaptation processes such as item collection reordering, menu contextualization, content priorization or user interface enrichment.
– Automatic binding of the data coming from the back-end services. This functionality acts as the glue between the services back-end and the front-end.
– Execute the application flow according to the author's intentions and taking into account access mechanism restrictions.

The Delivery Access Layer should be guided in order to operate in accordance to the developer's / author's intentions. Therefore, a declarative language capable of expressing different "Adaptation Policies" should be available to drive this layer.

## 4.2 Workspace Layer Composition

Taking advantage of the results produced by Access Layer Development, Access Layer Composition allows end users to build their own instant applications. These applications can be composed by mashing up gadgets extracted from a NG-SOA Broker Gadget Repository. Gadget mashup should be construed generally not only as a spatial composition of gadgets inside a dynamic environment but also as the construction of fully integrated applications. To do so, several platform service modules are defined:

– Wiring Communication Module
– Session Module
– Knowledge Module
– Context Module
– User Preferences Module
– Identity Management Module

These modules are software packages oriented to support specific architectural objectives. They provide support for a set of standard gadget capabilities, so they can use platform functionality in a loosely coupled way. Each gadget should define a set of policies in order to negotiate the usage of available platform capabilities, because not every platform will necessarily have to include all the capabilities. For instance, support for the persistence capability can be optional

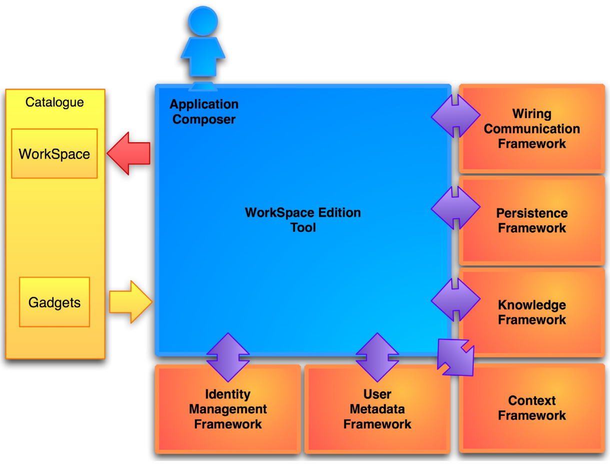


**Fig. 2.** Composition layer of the reference architecture

for one gadget and mandatory for another. The usage of the capabilities should be defined declaratively and will be the main link between a gadget and a specific mashup platform (however it is necessary to define a standard handler mechanism linking capabilities to gadget functionality)(see Fig. 2).

## 4.3 Access Layer Development

The next generation SOA access layer consists of composite applications that are based on service front-end resources or gadgets. These gadgets are the building blocks of the interface and behaviour to be designed and developed, and constitute single access elements to the underlying service back-end resources. These gadgets are self-contained components focused on a single goal. This reduces their complexity. Gadgets make up a substantial part of the interface and logic for invoking one or more use-case services, the user query forms and screens (see Fig. 3).

The gadget development process should be user-centric rather than program-centric, as the user is its final consumer. Thus the developers should bear users in mind when developing the front-end that they will see and interact with. This new approach calls for the use of fully visual development environments. Visual environments are more accessible and flexible for developers, who are domain experts but not necessarily technical users. These developers will visually compose a gadget from its building blocks (i.e. UI artifacts, back-end resources, operators), establish the connection to back-end Web Services, and integrate gadgets with the agnostic graphical user interface, thereby creating a service front-end resource.

The potential users of the next-generation SOA front-end should not be tied to any technology or access channel. This will promote the use of the wide range of services that will be available over the Internet. One of the objectives of

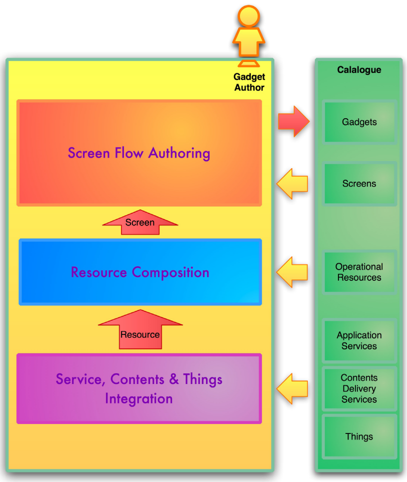


**Fig. 3.** Access layer development

the access layer is to serve as a multi-platform access mechanism to the SOA, suitable not only to be accessed by any platform or technology used at the client (desktop PC, mobile device, etc.) but also independently of the deployment infrastructure (i.e. the service front-end workspace platform). This technology independence should also be considered in the design and implementation of the gadgets that will make up the service front-end access layer. As a result, a device- and mashup platform-independent authoring language is needed to define the gadgets (i.e. their behaviour and interface) in a way that can be automatically converted to the target service front-end workspace platform and device technology. The gadgets will be defined in this language agnostically, with no focus on any specific access mechanism, and following a visual approach as mentioned above. The development environment is in charge of maintaining the mappings between the different layers of the authoring language.

Gadgets are not just mere UI parts and behaviour; they can form more sophisticated functionality around a piece of coherent business workflow. Gadget complexity can therefore vary depending on whether the UI comprises a single screen (i.e. simple gadget) or a screenflow inside the gadget itself (i.e. a complex gadget). This idea is especially relevant for highly constrained devices with limited display capabilities, such as mobile devices. Mashup-based compositional applications are seldom well designed for these devices because the user experience is based on disparate simultaneous views that are not affordable on small screens. The use of screenflows to model business logic in a single-piece view is better suited for these devices. The use of complex service front-end resources or complex gadgets in mobile environments is a good example of taking a proactive environment-dependent attitude, which, like technology independence, is a key objective of the SOA service front-end layer. These context-aware gadgets are then programmed to adapt their behaviour, interface and, maybe, part of their functionality to the specific context in which they have been deployed and are running.

## 5 Related Work and Future Trends

The approach depicted in this paper is being developed as part of the NEXOF-RA initiative[1], a project partially funded under the European Commission's 7th Framework Programme. NEXOF-RA is contributing to NESSI, the networked European software and services initiative. This initiative asserts that Information and Communication Technology (ICT) will be an essential driving force for innovation and a core enabler of economic growth in the coming years. Enterprises in Europe (both private and public sector) are facing significant structural changes and will rely on software and services to support them in adapting effectively. The main focus of NESSI is that of service and its overall ambition is to deliver NEXOF, a coherent and consistent open service framework leveraging research in the area of service-based systems to consolidate and trigger innovation in service-oriented economies. NEXOF-RA is the specific effort to define a model and a reference for this open service framework.

In addition, there are several projects, led by some of the authors of this paper, related to the approach of this paper and that are very close to NESSI's objectives and ambitions:

– MyMobileWeb[2] is the open source reference implementation of the next generation content and application adaptation platform for the mobile web. MyMobileWeb enables the (time-to-market) creation of high quality mobile applications capable of adapting to multiple delivery contexts.
– EzWeb[3] pursues the development of an enriched enterprise mash-up platform and the development of key technologies to be employed in building the front-end layer of new generation SOA architecture.
– FAST[4] aims at providing an innovative visual programming environment that will facilitate the development of next-generation composite user interfaces. It is a novel approach to application composition and business process definition from a top-down user-centric perspective.

Future work will focus on evolving the proposed reference architecture, as an open source service framework that builds on all the key guiding principles described above and on the proposed vision of the Internet of Services. We expect this architecture to become a major hub for the publishing, brokerage, customization and finally the consumption of Web-based resources on a global, cross-organizational scale, revolutionizing the user-service interaction.

[1] NEXOF-RA project, http://www.nexof-ra.eu/
[2] MyMobileWeb Project, http://mymobileweb.morfeo-project.org
[3] EzWeb Project, http://ezweb.morfeo-project.org
[4] FAST Project, http://www.fast-project.eu

## 6 Conclusion

The appearance of user-centric approaches to next generation service front-ends, such as the one proposed in this paper, will be a major step forward, providing solutions to currently hard-to-solve problems in the traditional SOA paradigm. The emergence of such service architectures will solve key problems in three different scenarios. Large enterprises may capitalize on faster application development (for what are known as instant applications), a more agile system landscape and the empowerment of their employees to design their own applications that best satisfy their unique requirements, and to share this knowledge with other employees better than in traditional Web service architectures.

On the other hand, the proposed architecture enables SMEs to find, customize, combine, catalogue, share and finally use applications that exactly meet their individual demands by leveraging the SaaS model, viewed as Utopian from a traditional SOA perspective. Supported by the new Internet of Services approach, they can select and combine resources hosted by third parties rather than buying a pre-determined, inflexible and potentially heavyweight solution or deal with complex B2B services.

Finally, individuals benefit from a sharp increase in the potential for personalization and participation. This approach will provide end-users with intuitive, unsophisticated IT ways to discover, remix and use those Web-based services that they consider interesting and useful. It will also allow them to participate, swap information with other users and service providers and to actively contribute in a way that encourages extensive use of the resources offered. This speeds up the service innovation pace. Focusing on the "long tail" advanced by Chris Anderson rather than a limited number of sophisticated experts, a user-centric SOA will involve the bulk of private users or small businesses and allow for "customer self-service".

## Acknowledgments

This work is supported in part by the European Commission under the first call of its Seventh Framework Program (NEXOF-RA and FAST Projects, grant FP7-216446 and INFSO-ICT-216048) and by the European Social Fund and both the UPM and CAM under their respective Researcher Training programmes.

## References

1. Alonso, G., Casati, F., Cuno, H., Machiraju, V.: Web Services Concepts, Architectures and Applications. In: Data-Centric Systems and Applications. Springer, Heidelberg (2004)
2. McAfee, A.P.: Enterprise 2.0: The dawn of emergent collaboration. MIT Sloan Management Review 47(3), 21–28 (2006)
3. Hierro, J.J., Janner, T., Lizcano, D., Reyes, M., Schroth, C., Soriano, J.: Enhancing user-service interaction through a global user-centric approach to soa. In: Proceedings of The Fourth International Conference on Networking and Services (ICNS 2008), pp. 194–203. IEEE Computer Society Press, Los Alamitos (2008)

4. Schroth, C., Christ, O.: Brave new web: Emerging design principles and technologies as enablers of a global soa. In: Proceedings of the IEEE International Conference on Services Computing, 2007. SCC 2007, pp. 597–604 (2007)
5. Soriano, J., Lizcano, D., Cañas, M.n., Reyes, M., Hierro, J.J.: Fostering innovation in a mashup-oriented enterprise 2.0 collaboration environment. System and Information Science Notes 1(1), 62–69 (2007); SIWN International Conference on Adaptive Business Systems (ICABS 2007). Chengdu, China
6. Schroth, C., Janner, T.: Web 2.0 and soa: Converging concepts enabling the internet of services. IT Professional 9(3), 36–41 (2007)
7. Smith, R.: Enterprise mashups: an industry case study. In: Keynote at New York PHP Conference and Expo. (June 2006)
8. Lizcano, D., Soriano, J., Fernández, R., López, J.A., Reyes, M.: Tackling composite application developments from an enterprise 2.0 mash-up perspective. In: Proceedings of the 14th International Conference on Concurrent Enterprising (ICE 2008) (June 2008)
9. Gartner: Hype cycle for software as a service. Gartner research, Gartner Inc. (August 2006)
10. Anderson, C.: The Long Tail: Why the Future of Business Is Selling Less of More. Hyperion (July 2006)
11. Dey, A.K.: Understanding and using context. Personal and Ubiquitous Computing Journal 5(1), 4–7 (2001)
12. Fielding, R.T.: Architectural styles and the design of network-based software architectures. Ph.D thesis, University of California, Irvine (2000)

# Domain-Specific Languages for Service-Oriented Architectures: An Explorative Study

Ernst Oberortner, Uwe Zdun, and Schahram Dustdar

Distributed Systems Group, Information Systems Institute,
Vienna University of Technology, Vienna, Austria
{e.oberortner,zdun,dustdar}@infosys.tuwien.ac.at

**Abstract.** Domain-specific languages (DSLs) are an important software development approach for many service-oriented architectures (SOAs). They promise to model the various SOA concerns in a suitable way for the various technical and non-technical stakeholders of a SOA. However, so far the research on SOA DSLs concentrates on novel technical contributions, and not much evidence or counter-evidence for the claims associated to SOA DSLs has been provided. In this paper, we present a qualitative, explorative study that provides an initial analysis of a number of such claims through a series of three prototyping experiments in which each experiment has developed, analyzed, and compared a set of DSLs for process-driven SOAs. Our result is to provide initial evidence for a number of popular claims about SOA DSLs which follow the model-driven software development (MDSD) approach, as well as a list of design trade-offs to be considered in the design decisions that must be made when developing a SOA DSL.

## 1 Introduction

Service-oriented Architectures (SOA) use platform-independent interfaces, or services, for performing business processes [6]. A SOA, in which services realize individual process steps or tasks, is called a *process-driven SOA* [26]. Process-driven SOAs deal with multiple concerns, such as orchestration of business processes, information in processes, collaboration between processes and services, data, transactions, human-computer interaction, service deployment, and many more. Hence, many domains need to be considered. Furthermore, a SOA has different stakeholders, including various domain experts and technical experts [24].

Using Domain-Specific Languages (DSL) for SOAs, based on Model-driven Software Development (MDSD) [7,18], enables technical experts and domain experts to work at higher levels of abstraction compared to using technical interfaces, executable process models, or service interface descriptions, such as programming APIs, Business Process Execution Language (BPEL) code, or interfaces described in the Web Service Description Language (WSDL). Furthermore, MDSD can be used for separating concerns. Hence, the multiple concerns of process-driven SOAs can be modeled independently through MDSD.

This paper presents an explorative study in which we developed a number of MDSD-based DSL prototypes, as well as a model-driven infrastructure to generate a running

© Springer-Verlag Berlin Heidelberg 2008

process-driven SOA from the models expressed in the DSLs. We present three experiments, in which we have focused on finding design decisions and/or trade-offs for developing model-driven DSLs. DSLs for domain experts (from now on called *high-level DSL*) and DSLs for technical experts (from now on called *low-level DSL*) were designed and developed. Two of our experiments deal with model-driven DSLs developed for process-driven SOAs, and the third one focuses on extending SOAs with Web user interfaces (UI), i.e., non-process-driven SOA concerns.

An in-depth study of specific claims about model-driven DSLs for SOAs was conducted. The claims target on (1) a systematic development approach for model-driven DSLs for SOAs, (2) the multiple concerns of SOAs, (3) the different levels of abstraction presented to the different stakeholders, and (4) on providing facilities for extensions. An analysis of the claims was made for each experiment in order to collect evidences and counter-evidences for claims about model-driven DSLs for SOAs. Hence, our results aim to help in making design decisions and considering the relevant design trade-offs, when engineering a model-driven DSL for SOAs.

This paper is organized as follows: Section 2 gives some background information on MDSD. Next, Section 3 describes our research method and discusses in detail the previously mentioned claims. Section 4 provides a description of our research experiments. The observations and results of the experiments are discussed in Section 5. Section 6 compares to related work. Finally, Section 7 concludes the paper.

## 2 Background: DSLs in Model-Driven Software Development

In the initial phase of our study we decided to focus on the MDSD approach to develop DSLs for SOAs (details about the reasons can be found in Section 3). Before we go into details of the technical parts of our study, we want to briefly explain the background.

MDSD is based on the notion of DSLs or specification languages for modeling various types of models. DSLs are small languages that are tailored to be particularly expressive in a certain problem domain. The DSL describes knowledge via a graphical or textual syntax which is tied to domain-specific modeling elements through a precisely specified language model. That is, the DSL elements are defined in terms of the language model that can be instantiated in concrete application model instances. The application model instances are defined in the DSL's concrete syntax which represents the language model. The OMG's MDA proposal [16] is one specific MDSD approach that has some notable differences to our MDSD approach in general – especially in its sole focus on interoperability and platform independence.

An MDSD tool introduces some way to specify transformations. There are different kinds of transformations, such as model-to-model or model-to-code transformations. Also, different ways to specify transformations, such as transformation rules, imperative transformations, or template-based transformations, exist. In any case, the ultimate goal of all transformations in MDSD tools is to generate code in executable languages, such as programming languages or process execution languages. The MDSD tools are used to generate all those parts of the executable code which are schematic and recurring, and hence can be automated.

## 3 Research Method and Approach Overview

Many DSLs for specific aspects of SOAs have been designed (see for instance [15,10]), but to the best of our knowledge, no study provides evidence for specific aspects and claims associated to SOA DSLs. Hence, this research field is clearly of explorative nature. For this reason, we have decided to use an explorative, qualitative research method to get insights and evidences in this first study, following a similar approach to constructing a grounded theory [20]. Our plan is to use the results of this study in our future research to conduct more detailed qualitative and quantitative studies about specific aspects of our results.

In our case, the initial analysis has been performed by developing a number of DSLs in various projects (among others we considered those reported in [9,25]), as well as a thorough literature review and discussions with both experienced and new DSL developers. There are many ways to implement a DSL, such as using MDSD or extending a dynamic language (see [8] for details). Following our first results, we decided to investigate further on a specific kind of DSL development style: DSLs developed using MDSD (as proposed in [18,10]). We have decided for this style because, in our experience, the explicit support for language models is useful for representing the various concerns and stakeholders of a process-driven SOA. However, this decision limits the generalizability of the results of our study: not necessarily the results are applicable without modification for other styles of developing SOA DSLs.

After the initial investigation phase, we decided to conduct an in-depth study of specific claims associated to MDSD-based SOA DSLs using a controlled series of three prototyping experiments. In each experiment, we have developed a number of MDSD-based DSL prototypes, as well as a model-driven infrastructure to generate a running process-driven SOA from the models expressed in the DSLs. All three prototyping experiments have been conducted in a project that has run for 12 month and included 4 developers. Two developers worked with approximately 50% of their time for the full project duration, one contributed 20% of his time for the full duration, and one contributed approximately 50% of his time for 5 month. The project did not only include DSL development, but also development of other artifacts, such as models and transformations, needed to obtain a running prototype solution.

In particular, we investigated the following claims in-depth:

– Developing model-driven DSLs follows a systematic development approach [18,10].
– A process-driven SOA encompasses multiple concerns, such as orchestration of business processes, information in processes, collaboration between processes and services, data, transactions, human-computer interaction, service deployment, and many more. To express these concerns, it is claimed that using DSLs and language models reduces the complexity of the overall system, compared to a system developed without DSL/MDSD support [3].
– Using DSLs and language models for expressing SOA concerns enables developers and other stakeholders to work at a higher level of abstraction compared to using technical interfaces, such as programming APIs, executable process models expressed in BPEL code, or service interface descriptions such as WSDL (see

[24]). Hence, DSLs can be tailored by providing constructs that are common to the domain the different stakeholders work in [4]. This enhances the readability and understandability of each DSL for the different stakeholders. But, the different levels of abstractions imply the definition of integration points or transformations between the constructs of the DSLs from the different layers [5].
- Due to the different levels of abstraction, it is claimed that language models should provide clear extension points for integrating new concerns [19].

In our study, we performed three controlled experiments, in which a number of DSL prototypes have been developed:

- Realization of a number of DSLs for process-driven SOA basic concerns (basic concerns)
- Extension with additional DSLs for supporting long-running transactions and human participation (extensional concerns)
- Realization of DSLs for non-process-driven SOA concerns: extensions of process-driven SOAs with Web applications, especially Web UIs (external concerns)

Step-by-step we analyzed the various claims by reviewing and analyzing the design decisions made in our project. Within each experiment, we compared the different DSLs and their artifacts (such as DSL syntax, language models, transformations, and extension points) and used the results as input for our study. Also, the inputs led to refactoring of the DSLs in order to improve them. In addition, with each additional experiment stage, we compared the DSLs between the stages. That is, we followed a constant comparison method, as advocated by grounded theory approach [20], throughout our study. For comparison, we used different methods, such as expert reviews of our DSLs and models, student experiments with the models, and the application of the DSLs and models in industrial case studies.

## 4 Study Details

Figure 1 outlines a systematic development approach of an MDSD-based DSL architecture, as proposed in [14,18,19]. High-level and low-level DSLs represent appropriate language models. Language models can have multiple DSL syntaxes. Furthermore, language models can have multiple language model instances, which are defined using the DSL's syntax. High-level DSL syntaxes, language models, and model instances extend low-level DSL syntaxes, language models, and model instances respectively. Low-level DSLs provide constructs that are tailored for technical experts, whereas high-level DSLs are tailored for domain experts. A suitable separation of concerns can be established by splitting the language model into high- and low-level models, where the high-level model extends the low-level model. Hence, a separation of technical and domain concerns can be established to present only the appropriate concerns to each of the different groups of stakeholders.

In this approach, high-level concerns, relevant for non-technical stakeholders, are distinguished from low-level technical concerns to achieve better understandability for the different stakeholders. Due to the diverse backgrounds and knowledges of the different

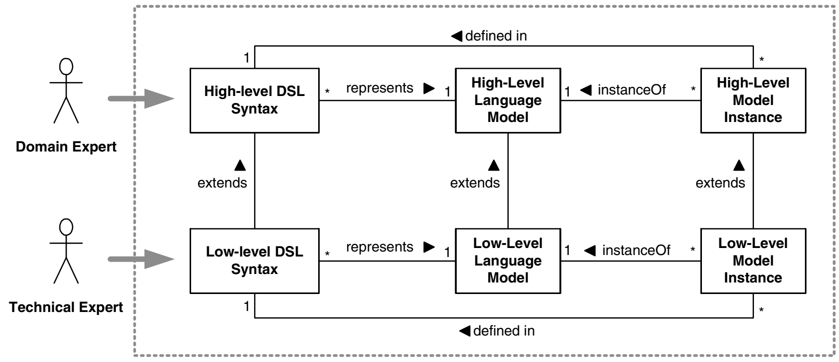


**Fig. 1.** MDSD - DSLs

stakeholders, it makes sense to present to each group of stakeholders only the models they need for their work, and omit other details, as proposed in [24]. That is, high-level DSLs, designed with support for the domain experts, enable to work in a language in which domain concerns are depicted in or close to the domain's terminology. For instance, in the banking domain terms like account, bond, fund, or stock order are used in the high-level DSL. Low-level DSLs, in contrast, are utilized by technical experts to specify the technical details missing in the high-level DSLs. These details are needed by the model-driven code generator to turn the model instances, expressed in the DSLs, into a running system. For instance, in the process-driven SOA domain, relevant low-level concerns are service, service deployment, process variable, or database connection.

In the field of this study, we tried to provide evidence or counter-evidence for the claims summarized above. In particular, we evaluated the claims for three experiments. The experiments deal with process-driven SOAs, as well as an extension of process-driven SOAs with Web UIs. We focused in our experiments on the design decisions made and on the design trade-offs that have been considered. At first we will to describe the experiments in detail and after that in Section 5 our main results.

In the first experiment, the language models were designed together and at the same time. The extension points were specifically designed for integrating the language models. The organization of the language models is shown in Figure 2(1). A Core language model provides the extension points for modeling the basic concerns of process-driven SOAs, such as collaboration, controlflow, and information. During the second experiment, the extension points were used for introducing extensional concerns for which the extension points in the basic models have not originally been designed for. The language model structure of the second experiment is shown in Figure 2(2). In the third experiment, we investigated in how far external extensions, i.e., non-process-driven SOA concerns, can be integrated with the existing language models for process-driven SOAs. In particular, we integrated Web UIs with the process-driven SOA models. The organization of the Web UI's language models is depicted in Figure 2(3).

*Process-driven SOA Basic Concern Language Models*
The first experiment concentrates on basic concerns of process-driven SOAs, as well as providing high- and low-level DSLs for the different stakeholders [4]. The View-based

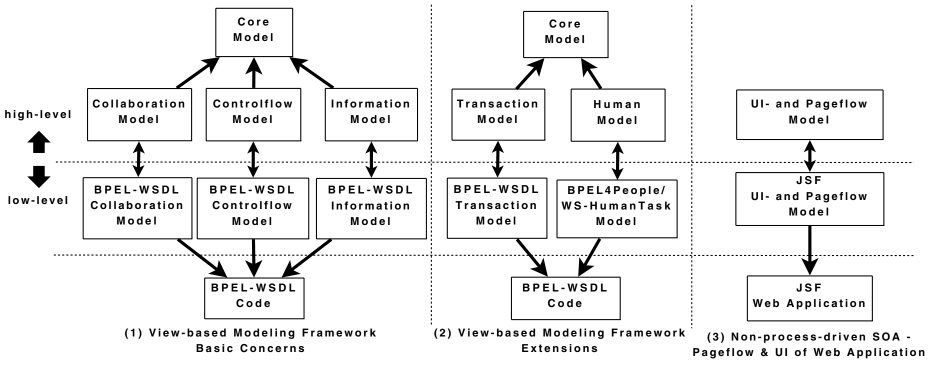

Core Model
Collaboration Model
Controlflow Model
Information Model
high-level
low-level
BPEL-WSDL Collaboration Model
BPEL-WSDL Controlflow Model
BPEL-WSDL Information Model
BPEL-WSDL Code
(1) View-based Modeling Framework Basic Concerns
Core Model
Transaction Model
Human Model
BPEL-WSDL Transaction Model
BPEL4People/ WS-HumanTask Model
BPEL-WSDL Code
(2) View-based Modeling Framework Extensions
UI- and Pageflow Model
JSF UI- and Pageflow Model
JSF Web Application
(3) Non-process-driven SOA - Pageflow & UI of Web Application

**Fig. 2.** Experiments Overview

Modeling Framework (VbMF) [22,23] is a model-driven framework for reducing the development complexity in process-driven SOAs, as well as to improve interoperability and reusability of models. It provides multiple language models, high-level and low-level, each responsible for a different concern of process-driven SOAs (i.e., controlflow, collaboration, and information). The structure of high-level and low-level language models is shown in Figure 2(1).

In this experiment, we used a systematic development approach as follows. First, high-level language models were designed. A central Core language model provides extension points for defining new language models for the appropriate concerns. Furthermore, it provides extension points for various language models for basic and extensional concerns. The following language models extend the Core language model for modeling basic concerns of process-driven SOAs:

- The **Controlflow** language model offers constructs for modeling controlflows of business processes, which consist of many activities and control structures. Activities are process tasks, e.g., service invocations or data handling. The execution order of activities is described through control structures, e.g., conditional switches.
- To compose the functionality provided by services or other processes, the **Collaboration** language model is used. This language model extends the Core language model to represent interactions between a business process and its partners.
- The **Information** language model represents the flow of data objects inside the business process. Furthermore, it provides a representation of message objects traveling back and forth between the process and the external world.

For each high-level language model, except the Core language model, the low-level language models were designed as an extension of the high-level language models. Both the high-level and low-level language models are close to the concepts of BPEL and WSDL. Finally, the DSL syntaxes were developed. The high-level DSL syntaxes are based on the constructs of the high-level language models, whereas low-level DSL syntaxes are based on the constructs of the low-level language models. Hence, domain experts can – with the help of technical experts – use the high-level DSLs for modeling domain concerns, and technical experts can model technical concerns with the low-level DSLs.

*Process-driven SOA Extensions of VbMF*
In contrast to the first experiment, which analyzed the basic concerns of process-driven SOAs, this experiment uses the introduced extension points of the Core language model for integrating extensional concerns, such as transaction and human language models. The goal of this experiment is to figure out if the systematic development approach, used in the first experiment, can be applied for extensional concerns of process-driven SOAs. The structure of the high- and low-level language models for both experiments is shown in Figure 2(2).

To extend VbMF for long-running transactions, transactional concerns were integrated into the VbMF through a newly defined language model [22,23]. In the same way as the Controlflow, Collaboration, and Information language models were created, the first step was to design a high-level **Transaction** language model which extends the Core language model. Afterwards, a low-level Transaction language model was designed which extends the high-level Transaction language model. Like the low-level language models of the first experiment, the low-level Transaction language model is based on BPEL and WSDL concepts too. Finally, high-level and low-level DSLs were developed to support the modeling of transactions, based on the constructs of the appropriate language model.

A second extension of VbMF is the support of human participation in SOA-based business processes [21]. Again, a high-level **Human** language model was designed which extends the Core language model. Human aspects are assigned to processes and activities. A low-level Human language model extends the high-level Human language model, and it is based on concepts from BPEL4People [2] and WS-HumanTask [1]. Finally, high- and low-level DSLs were implemented, based on the appropriate language models, to support the modeling of human tasks for SOA-based business processes.

*Non-Process-Driven SOA Extensions: Web User Interfaces*
The third experiment followed again the systematic development approach, as adopted in the first two experiments. The language model hierarchy is depicted in Figure 2(3). The goal of this experiment is to figure out, if the systematic development approach can also be applied to extensions of process-driven SOAs with non-process-driven SOA concerns. The experiment deals with the modeling of Web UIs for Web pages, as well as process-oriented modeling of the pageflow through Java-like IF-ELSE statements. Web UIs contain the input and output components which are displayed to the user on Web pages. The pageflow provides the basis for selecting the subsequent Web page that should be displayed to the user, dependent on the current page and user interactions, e.g., which link or button is pressed by the user.

First, the high-level language model is introduced for modeling the pageflow and the UIs of the Web pages. A low-level language model for modeling the pageflow is introduced which is based on the pageflow definition of JavaServer Faces (JSF) [11] Web applications. The DSLs were implemented to provide suitable modeling of the pageflow and the UIs. The developed DSLs provide constructs that are very similar to the language model. In this experiment, there is no need for a mapping between the constructs of the DSL and the constructs of the language model.

## 5 Study Results

In this section, we summarize the evidences and counter-evidences we found in our explorative experiments. First, the experiments provided some useful insights into design decisions required during the design of model-driven DSLs for process-driven SOAs:

- A design decision for the relation between DSL syntax and language model constructs must be made. We observed that in all three experiments the relationship between the names used in the DSL syntaxes and the names of the constructs defined in the language models was a concern. In all three experiments, we decided that the DSL syntaxes provide constructs that are named equivalently to the constructs in the language model. If the DSL syntax constructs are not named equivalently to the language model constructs, a more complex mapping between DSL and language model constructs is required, which means that extra efforts are required to develop this mapping. The mapping might also make the relationships between syntax constructs and models harder to understand. However, with a different naming in models and syntaxes, the syntax and modeling elements can be tailored more easily.
- In all three experiments, the low-level language models are extensions of the high-level language models. Hence, a relationship exists between them. A design decision must be made, in which order and dependency the high-level and low-level models are designed. The high-level language models can be designed first, and afterwards the low-level language models. Hence, domain concerns can be expressed close to their domain notions, such as compliance concerns in business processes. Another possible design approach is to derive the high-level language models from the low-level language models, which are based on technical concerns, e.g., constructs similar to BPEL (as done in our basic models). In this case, emphasis must be put on the high-level design of technical concerns, in order to make them understandable to domain experts, too. This is often not easy. Yet another approach is to design high- and low-level language models and DSLs in parallel. The main problem lies in the huge differences of the offered constructs between the languages. An example are languages like the Business Process Modeling Notation (BPMN) and BPEL. This approach requires a mapping between the often incompatible high-level and low-level language models, with possible inconsistencies. A part of this design decision is the development order of the high- and low-level language models and DSL syntaxes. If possible, the design of the high-level DSL syntax and language models should be performed together with the domain experts.
- In the first two experiments, which deal with basic and extensional concerns of process-driven SOAs, multiple language models where used. Multiple language models reduce the complexity by separation of concerns. This leads to providing tailored views for the different stakeholders. The main challenge of splitting lies in finding appropriate extension points for merging models. Poor extension points can lead to inconsistencies between the models. In addition, merging through extension points is more complex than using modeling abstractions, such as associations. In the third experiment, one language model is used for modeling the pageflow and UIs of Web applications. Having only one language model does not provide a good

separation of concerns for the development team and other stakeholders, but, on the other hand, there is no need for providing suitably designed extension and integration points, as well as possibly complex merging algorithms for the integration of multiple models. The design decision is whether it makes sense to split one language model into multiple models or not, and if splitting is chosen, where to split. Trade-offs for this design decision concern the number of concerns, development teams, and stakeholders.

Second, we found the following evidences for model-driven DSLs for claims associated to Section 3:

– It is possible to follow a systematic development approach, such as the one described in our three experiments in Section 4. In our case, this is not only valid for process-driven SOAs but also for non-process-driven SOA concerns, such as in our case Web applications.
– The systematic development approach used for the basic concerns of process-driven SOAs, such as controlflow, collaboration, or information of process-driven SOAs, can be followed for modeling extensional concerns, such as the transactional or human concerns in our experiments.
– Through a separation in high- and low-level DSLs, it is possible to support different stakeholders with different background and knowledge, i.e., domain experts and technical experts.
– Model-driven DSLs can enhance the understandability and readability for the individual stakeholders of a process-driven SOA. Furthermore, MDSD-based DSLs can reduce the complexity of process-driven SOAs.

Finally, the following counter-evidences, for the claims described in Section 3, should be considered as design trade-offs for the development of model-driven DSLs for SOAs:

– It is possible that the integration of high- and low-level concerns lead to DSL language design issues, such as redundancy in languages, inconsistencies, and which language should be chosen for overlapping concerns.
– Detailed separations of one language model into multiple ones can result in loose coupling of the different language models. Thus, the result is: the more detailed the separation, the more complex the model integration points for merging the different application models. Possible ways to achieve model integration are name-based matching, ontology-based matching, or inheritance. Hence, there is a trade-off between the complexity of the integration points and the degree of separation of concerns achieved in the language models.
– We observed another trade-off between model integration point design for the different stakeholders and the understandability, as well as the readability. The more complex the integration points are, the less understandable and readable the DSLs and/or their language models get in many cases. Hence, enhancing understandability and readability for one type of stakeholders increases the complexity of integrating models for other stakeholders. That is, the complexity for stakeholders, who need to integrate and understand all models at once, can rise even though the complexity for individual stakeholders decreases.

## 6 Related Work

Pitkänen and Mikkonen [17] argue that well designed DSLs, modeling tools, and code generators increase the productivity. They concentrate on lightweight and modular DSLs instead of full-blown DSLs. Some situations of full-blown DSLs are described, e.g., several different implementation platforms. The lightweight approach can be an aid in defining the scope and concepts of DSLs before the implementation of a full-blown DSL starts. In comparison to our study, the systematic development approach can be applied to lightweight, as well as full-blown DSLs. The different design decisions and/or trade-offs, described in Section 5, are also valid for developing lightweight model-driven DSLs.

Bierhoff et al. [13] describe an incremental approach for developing DSLs. First, they choose an application and develop a DSL which is expressive enough to describe the application. Also, domain boundaries are defined. Then, the DSL grows until it is too expensive to extend it more. The approach is demonstrated on CRUD applications, i.e., create, retrieve, update, delete applications. The approach by Bierhoff et al. reflects the evolution of our three experiments described in Section 4. Also, we started by an initial experiment and extended it incrementally.

Maximilien et al. [15] developed a DSL for Web APIs and Services Mashups. A number of interesting design issues for DSLs are mentioned: (1) levels of abstraction, (2) terse code, (3) simple and natural syntax, and (4) code generation. These goals are very similar to our proposed claims. The developed DSL is used for SOAs, and the described approach and results are in line with our results.

Tolvanen [12] provides a guidance for defining and developing DSLs based on his long-year experiences in building DSLs. The development process is divided into four phases: (1) Identifying abstractions, (2) specifying the language models, (3) creating notations for the language based on the language models, and (4) defining model validators and code generators. The development phases are very similar to our observations. We started by defining abstractions of the domain, designed high- and low-level language models, developed a DSL with notations equivalent to the language models. Also, we provide model validators and code generators. The proposed approach by Tolvanen is similar to our systematic development approach for model-driven DSLs: (1) identifying the concepts of the domain and their relations, (2) designing the language models, (3) developing the DSLs based on the language models, and (4) generating code of valid domain models through a code generator.

## 7 Conclusion

The scope of this paper is to provide an initial study on a systematic development approach for SOA DSLs based on MDSD. It is likely, but not necessary, that many of our results are also valid for other DSL implementation techniques. We followed the MDSD-based DSL approach quite closely. Hence, our results should be valid for a wide range of DSL tools.

We have addressed a broad range of process-driven SOA concerns (including basic, extensional, and external concerns). One result is that all of them can be expressed relatively easy using a distinct language model and integrated with existing language models using simple techniques such as inheritance or matching algorithms. However, it is possible that other process-driven SOA concerns exist for which this is not easily possible.

Also, this paper discusses the design decisions and/or trade-offs we observed, as well as evidences and counter-evidences for the claims around model-driven DSLs for SOAs. Model-driven DSLs can enhance complexity, understandability, and readability for the individual stakeholders of SOAs. Therefore, DSLs can be tailored for domain experts and technical experts. But enhancing understandability and readability for domain experts, decreases understandability and readability for technical experts and vice versa.

During our study, we have used only a limited number of comparison and analysis techniques (such as code reviews, expert reviews, and student experiments). Other comparison or analysis methods might reveal properties that have not been revealed so far with our techniques used. Hence, we want to follow the systematic development approach in more studies and analyze the results.

## Acknowledgement

This work was supported by the European Union FP7 project COMPAS, grant no. 215175.

## References

1. Agrawal, A., Amend, M., Das, M., Keller, C., Kloppmann, M., König, D., Leymann, F., Müller, R., Pfau, G., Ploesser, K., Rangaswamy, R., Rickayzen, A., Rowley, M., Schmidt, P., Trickovic, I., Yiu, A., Zeller, M.: Web Services Human Task (WS-HumanTask), version 1.0 (2007)
2. Agrawal, A., Amend, M., Das, M., Keller, C., Kloppmann, M., König, D., Leymann, F., Müller, R., Pfau, G., Ploesser, K., Rangaswamy, R., Rickayzen, A., Rowley, M., Schmidt, P., Trickovic, I., Yiu, A., Zeller, M.: WS-BPEL extension for people (BPEL4People), version 1.0 (2007)
3. Jansen, A., Bosch, J.: Software Architecture as a Set of Architectural Design Decisions. In: WICSA 2005: Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture, Washington, DC, USA, pp. 109–120. IEEE Computer Society, Los Alamitos (2005)
4. Schmidmeier, A.: Aspect oriented DSLs for business process implementation. In: DSAL 2007: Proceedings of the 2nd workshop on Domain specific aspect languages, p. 5. ACM, New York (2007)
5. Sánchez-Ruíz, A.J., Saeki, M., Langlois, B., Paiano, R.: Domain-Specific Software Development Terminology: Do We All Speak the Same Language?, http://www.dsmforum.org/events/DSM07/papers/sanchez-ruiz.pdf
6. Barry, D.K.: Web Services and Service-oriented Architectures. Morgan Kaufmann Publishers, San Francisco (2003)
7. Fowler, M.: Language workbenches and model driven architecture (June 2005), http://www.martinfowler.com/articles/mdaLanguageWorkbench.html
8. Fowler, M.: Language workbenches: The killer-app for domain specific languages? kbench (June 2005), http://www.martinfowler.com/articles/languageWorkbench.html

9. Goedicke, M., Koellmann, K., Zdun, U.: Designing runtime variation points in product line architectures: three cases. Science of Computer Programming 53(3), 353–380 (2004)
10. Greenfield, J., Short, K.: Software Factories: Assembling Applications with Patterns, Frameworks, Models & Tools. J. Wiley and Sons Ltd., Chichester (2004)
11. JavaServer Faces Technology, http://java.sun.com/javaee/javaserverfaces/
12. Tolvanen, J.-P.: Domain-Specific Modeling: How to Start Defining Your Own Language (last accessed, July 2008), http://www.devx.com/enterprise/Article/30550
13. Bierhoff, K., Liongosari, E.S., Swaminathan, K.S.: Incremental Development of a Domain-Specific Language That Supports Multiple Application Styles. In: OOPSLA 6th Workshop on Domain Specific Modeling, pp. 67–78 (October 2006)
14. Luoma, J., Kelly, S., Tolvanen, J.-P.: Defining Domain-Specific Modeling Languages: Collected Experiences. In: Proceedings of the 4th OOPSLA Workshop on Domain-Specific Modeling (DSM 2004), Vancouver, British Columbia, Canada (2004)
15. Maximilien, E.M., Wilkinson, H., Desai, N., Tai, S.: A domain specific-language for web apis and services mashups. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749, pp. 13–26. Springer, Heidelberg (2007)
16. OMG. MDA Guide Version 1.0.1. Technical report, Object Management Group (2003)
17. Pitkänen, R., Mikkonen, T.: Lightweight Domain-Specific Modeling and Model-Driven Development. In: OOPSLA 6th Workshop on Domain Specific Modeling, pp. 159–168 (October 2006)
18. Stahl, T., Voelter, M.: Model-Driven Software Development. J. Wiley and Sons Ltd., Chichester (2006)
19. Cook, S.: Domain-Specific Modeling and Model Driven Architectures (2004), http://www.bptrends.com
20. Strauss, A., Corbin, J.: Grounded theory in practice. Sage, London (1997)
21. Holmes, T., Tran, H., Zdun, U., Dustdar, S.: Modeling Human Aspects of Business Processes - A View-Based, Model-Driven Approach. In: ECMDA-FA, pp. 246–261 (2008)
22. Tran, H., Zdun, U., Dustdar, S.: View-based and model-driven approach for reducing the development complexity in process-driven SOA. In: Abramowicz, W., Maciaszek, L.A. (eds.) BPSC, GI. LNI, vol. 116, pp. 105–124 (2007)
23. Tran, H., Zdun, U., Dustdar, S.: View-based integration of process-driven soa models at various abstraction levels. In: Kutsche, R.-D., Milanovic, N. (eds.) Proceedings of First International Workshop on Model-Based Software and Data Integration MBSDI 2008, pp. 55–66. Springer, Heidelberg (2008)
24. Perrone, V., Bolchini, D., Paolini, P.: A Stakeholders Centered Approach for Conceptual Modeling of Communication-Intensive Applications. In: SIGDOC 2005: Proceedings of the 23rd annual international conference on Design of communication, pp. 25–33. ACM, New York (2005)
25. Zdun, U.: Tailorable language for behavioral composition and configuration of software components. Computer Languages, Systems and Structures: An International Journal 32(1), 56–82 (2006)
26. Zdun, U., Hentrich, C., van der Aalst, W.: A survey of patterns for service-oriented architectures. International Journal of Internet Protocol Technology 1(3), 132–143 (2006)

# Managing the Alignment between Business and Software Services Requirements from a Capability Model Perspective

Eric Grandry, Eric Dubois, Michel Picard, and André Rifaut

Public Research Centre Henri Tudor,
29, av. J. F.Kennedy, L-1855 Luxembourg, Kirchberg
{eric.grandry,eric.dubois,michel.picard,andre.rifaut}@tudor.lu

**Abstract.** In this paper we introduce a framework for capturing and managing the requirements associated with the non-functional part of the services like service management, security management, assurance, for which norms, recommendations and good practices exist. The proposed framework considers these service requirements both from a business and a software perspective. The elicitation, the capture and the traceability issues related to these requirements are solved with goal-oriented requirements engineering techniques, while the structuring and the assessment of the requirements is based on the ISO/IEC-15504 standard. The overall framework is illustrated with a business case run by our research centre in a public/private partnership. It is associated with the design of project management services delivered through a portal and is focusing on the services management requirements in relation with the IT service management ISO/IEC 20000 norm.

**Keywords:** Business Service Design, Service Level Objective, Capability Model, Goal-Oriented Requirements Engineering, Service management.

## 1 Introduction

Analogously to [1] we make a distinction between QoS associated with software services (like e.g. those considered in the ISO/IEC 9126 standard [2]) and those associated with the business facet of a service, i.e. that add value to a service at a value web level [3]: the SERVQUAL model [4] or the recent work of O'Sullivan et al. [5] consider credibility, trust, security, availability as business level attributes of QoS. Our view is in line with these approaches but focuses on a specific dimension: the capability of a service provider to offer a service that is compliant with a number of assurance and regulation reference models. Assurance reference models include ITIL [6], ISO 20000 series for service management [7], ISO 27000 series for security management [8]. Regulation reference models include Sarbanes-Oxley [9], COSO [10] or Basel II [11] in the financial sector. These reference models define rules and objectives that organizations need to comply. We claim that the capability to comply should be part of the services the organizations expose since it represents elements of value for the services consumers. We therefore propose a framework that answers the following research issues:

© Springer-Verlag Berlin Heidelberg 2008

1. How to identify, to capture and to structure the requirements on the business QoS underlying these assurance and regulation reference models. This is done by using the ISO/IEC 15504 standard [12] (abbreviated to "15504") that offers a template for organizing the requirements into a framework allowing to measure the capability and the performance level of an organization to comply with reference models.
2. How to support the alignment between the business perspective on assurance and regulatory requirements and the lower-level software services requirements implementing the business service.

To answer these two questions, we will use Goal-Oriented Requirements Engineering (GORE) models to support the formalization of QoS requirements as well as in support to their traceability at business and software levels. In the rest of the paper we will use i* [13]. However other GORE notations are also eligible.

We will illustrate the whole approach with a real business case currently managed by our public research centre together with a network of professional consultants in project management [14]: setting up a portal based on a SOA architecture offering project management services dedicated to SME's that usually do not have the resources (money and/or competences) to access complete project management software suites For the sake of this paper, we will focus only on one facet of the assurance requirements associated with these services: the IT service management. IT service management is handled through different norms, including ITIL [6] documented by the British Office of Government Commerce (OGC) and the emergent ISO/IEC 20000 [7] (abbreviated to "20000") organized in two parts under the general title of "Information Technology – Service Management".

In section 2, we present the role of GORE in the alignment between business and software services, and outline our proposed approach. In section 3 we detail the 15504-based approach applied to the structuring and the performance measurements of requirements associated with the IT 20000 service management. The section 4 explains our approach regarding the progressive refinement of business-oriented and software-oriented service management requirements. Finally section 5 wraps up the paper and discusses some open issues.

## 2 Requirements Engineering and Service Description

Figure 1 revisits the well-known business/IT alignment from [15] and introduces the role that GORE plays in guaranteeing the business/software services alignment thanks to the fact that goals can be used at different decision-making levels. At the bottom of the vertical axis, one can see the traditional use of GORE for the progressive elaboration of IS/software systems from strategic goals. Goals support the characterization of a system in terms of desired state of affairs to be achieved and/or maintained. GORE has been proven useful in the progressive elicitation and structuring of the requirements (usually referred as non-functional requirements – NFR) related to QoS. Such NFR can be attached e.g. to use cases associated with the description of software services. Several examples of this approach can be found in the literature using notations like those provided by KAOS [16] or *i** [13].

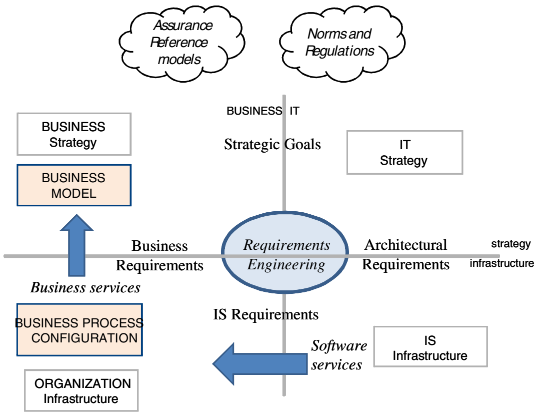


**Fig. 1.** GORE in support to Business/Software Services Alignment

On the left part of figure 1, we define business services from a business value model perspective: the functional business part of these services directly contributes to the creation of value in a networked value constellation according to the strategy defined by an organization [3]. We do not cover how these business services are discovered but we can refer to approaches as e3value [1] or strategic map [17]. Business strategy does not only include an economic and financial dimension but has to accommodate with constraints from the environment of the organization such as national, cross-industry or industry-specific regulations and assurance best practices. These constraints need to be formalized and structured in terms of requirements (top part of figure 1). In section 3, we present the joint application of GORE and 15504 to meet this objective.

Business services functional and non-functional requirements express specifications against which different solutions in terms of business processes (BP) configurations can be designed and evaluated (left bottom-part of figure 1). Our work does not concentrate on how to design such BP configurations, but once the BP's have been identified, they still need to be realized through an IT/IS (Information System) (right bottom part of figure 1). From a requirements perspective, this calls for a further refinement of business requirements into software requirements. In the domain of QoS attributes we need to refine non-functional business requirements derived from norms and assurance best practices into software requirements that can be functional and/or non functional. Section 4 will illustrate this aspect of our work.

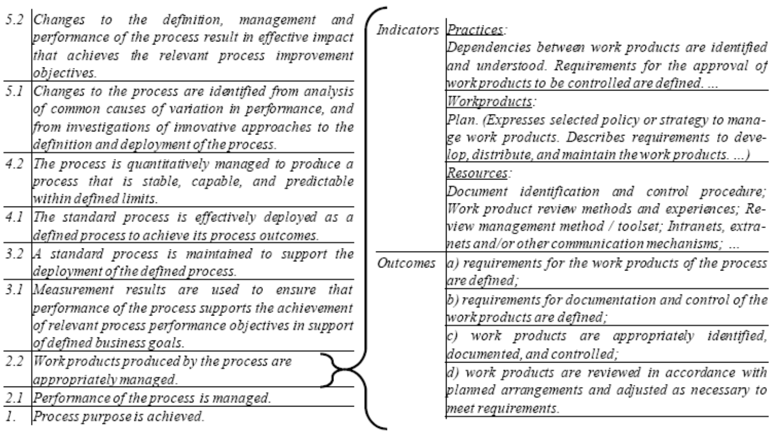
## 3 Elicitation and Structuring of Service Management Requirements

We use a methodology for the progressive elicitation, formalization and structuring of catalogues of QoS requirements inherent to regulations and assurance norms. This approach is based on a joint use of GORE and 15504 standard. The methodology has been extensively applied in our Centre for the design of different business requirements compliance frameworks in different domains as, for example: security management [18] (based on ISO 17799 and ISO 27000 series [8]), knowledge management [19], project management, credit management, venture capital management [20], risk management and Anti-Money Laundering compliance management for the fund industry and Basel II operational risk management [21]. In this section we present the results of the application of this methodology on the 20000 IT service management norm. This section summarizes the method defined in [30] and [24] and applied in [22]. In all these domains, the approach is always to capture requirements associated with the QoS of BP to be put in place and to define the required level of performance capability needed from the service provider in their execution.

## 3.1 The ISO 15504 Assurance and Performance Framework Model

For structuring and organizing the business QoS requirements inherent to regulations and assurance norms, we have found and experimented a valuable requirements template and associated guidelines that are made available through the 15504 standard [12]. In this 15504 a generic requirements' taxonomy and a predefined requirements' structure define a framework used for eliciting and structuring QoS requirements as well as for assessing and measuring the compliance of deployed BP against these requirements. Analogously to standards such as COSO [10] and CMM [23], 15504 (previously known as SPICE) provides an assessment model against which the *assurance* aspects of an organization in terms of realization of its BP and their contribution to business services objectives can be defined and measured. Built on top of those predecessors, the main originality of 15504 "Process Assessment Model" (PAM) is to standardize the structure of assurance requirements by defining a taxonomy of generic BP assurance goals that are applicable to BP of business domains not limited to IT software engineering domain. Figure 2 presents the generic guidelines associated with the construction of a PAM. On the left part of figure 2, from the bottom to the top, one can read the business capability goal of the services at level 1, and then, from 2.1 to 5.2, the different level of assurance that can be associated to this business goal.

According to 15504, a Process Assessment Model (PAM) describes requirements on BP implementing QoS assurance attribute with the purpose and outcomes of each assurance attribute. The **purpose** of an assurance attribute "*describes at a high level its overall objectives*" [12]. Each purpose is fully decomposed into **outcomes**. Each outcome is an observable achievement of some assurance attribute. Actually, an outcome describes that an artifact is produced, or that a significant change of state occurred, or that some constraint has been met. Outcomes can be further detailed with **indicators** focusing on "*sources of objective evidence used to support a judgment about the fulfillment of outcomes*", for instance: work products ("*an artifact associated with the execution of a process*"), practices ("*activities that contributes to the purpose or outcomes of a process*"), or resources (e.g. "*human resources, tools, methods and infrastructure*") [12].



| | |
|---|---|
| 5.2 | Changes to the definition, management and performance of the process result in effective impact that achieves the relevant process improvement objectives. |
| 5.1 | Changes to the process are identified from analysis of common causes of variation in performance, and from investigations of innovative approaches to the definition and deployment of the process. |
| 4.2 | The process is quantitatively managed to produce a process that is stable, capable, and predictable within defined limits. |
| 4.1 | The standard process is effectively deployed as a defined process to achieve its process outcomes. |
| 3.2 | A standard process is maintained to support the deployment of the defined process. |
| 3.1 | Measurement results are used to ensure that performance of the process supports the achievement of relevant process performance objectives in support of defined business goals. |
| 2.2 | Work products produced by the process are appropriately managed. |
| 2.1 | Performance of the process is managed. |
| 1. | Process purpose is achieved. |

| | |
|---|---|
| Indicators | Practices: Dependencies between work products are identified and understood. Requirements for the approval of work products to be controlled are defined. ... |
| | Workproducts: Plan. (Expresses selected policy or strategy to manage work products. Describes requirements to develop, distribute, and maintain the work products. ...) |
| | Resources: Document identification and control procedure; Work product review methods and experiences; Review management method / toolset; Intranets, extranets and/or other communication mechanisms; ... |
| Outcomes | a) requirements for the work products of the process are defined; |
| | b) requirements for documentation and control of the work products are defined; |
| | c) work products are appropriately identified, documented, and controlled; |
| | d) work products are reviewed in accordance with planned arrangements and adjusted as necessary to meet requirements. |

**Fig. 2.** GORE in support to Business/Software Services Alignment

Outcomes and indicators are organized into different aspects. The first aspect is related to the main activity while the other aspects are related to different assurance aspects associated with the activity. This results in a taxonomy of assurance requirements goals. The right part of Fig 2 lists the different aspects and details the objectives of the outcomes and indicators associated with the assurance aspect “2.2”.

| Service Level Management | 1 | 2.1 | 2.2 |
|---|---|---|---|
| **Purpose** | Service Level is defined and agreed with the service customer, and recorded and managed | … | The service level agreement is adequately managed |
| **Outcomes** | a) Service level is agreed on the basis of the customer needs and documented<br>b) Service level is monitored according to the agreed SLA<br>c) Service level are monitored and reported against targets | | a) SLA is standardised<br>b) SLA is reviewed internally |
| **Indicators** | Practices:<br>Agree SL; Monitor SL; Report SL<br>Work Products:<br>SLA; SL Report | | Practices:<br>Standardise SLA<br>Work Products:<br>Standardised SLA |

**Fig. 3.** Requirements associated with the Service Level Management QoS attribute

For ease of understanding, a concrete example (instantiation) is given in figure 3 with a fragment of the final result of our methodology applied to the 20000 IT Service Management document. The application of the methodology has resulted in the transformation of natural language flat requirements from the norm to structured requirements in a PAM. This work has been performed by a CRP Henri Tudor’s team in the context of a New Work Item accepted in the Sub Committee 7 of the ISO/IEC JTC1 (Joint Technical Committee on Information Technology) dealing with Software and Systems Engineering. More details about this application can be found in [24]. The presented fragment illustrates a part of the requirements associated with one QoS related to “Service Level Management”. In total 17 other QoS attributes have been characterized in terms of their associated requirements. They include e.g.: Incident Management, Problem Management, Change Management, Information Security management.

## 3.2 Building Compliant 15504 Service Management Requirements Models

As explained in the preceding section, 15504 helps to better structure goal-based QoS requirements models with PAM. Difficulties arise when creating those PAM: 15504 does not provide any guidance in the incremental elaboration of a PAM. It provides generic concepts used in PAM and rules (meta-requirements) that must be satisfied by PAM, but gives no guidance to the identification of the business processes, nor the formalization of the knowledge domain which is needed for that. This guidance can be given by GORE techniques, such as *i** [13] which relies on a taxonomy of concepts close and compatible to those of 15504. The rules and heuristics that we have discovered regarding the use of *i** in support to the progressive and systematic elaboration of PAM are presented in [24]. They are summarized in the next paragraph in the context of the elaboration of the paraphrased result presented in figure 3.

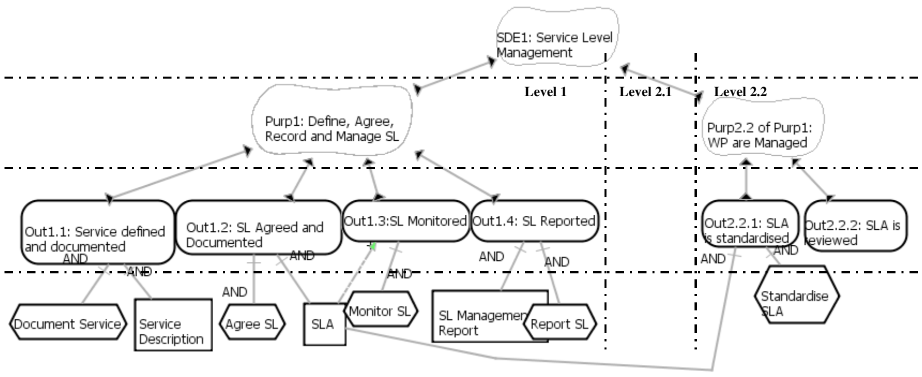

SDE1: Service Level Management
Level 1
Level 2.1
Level 2.2
Purp1: Define, Agree, Record and Manage SL
Purp2.2 of Purp1: WP are Managed
Out1.1: Service defined and documented
Out1.2: SL Agreed and Documented
Out1.3:SL Monitored
Out1.4: SL Reported
Out2.2.1: SLA is standardised
Out2.2.2: SLA is reviewed
AND
Document Service
Service Description
Agree SL
SLA
Monitor SL
SL Management Report
Report SL
Standardise SLA

**Fig. 4.** Requirements Goal Tree Associated with the Service Level Management Attribute

Let us now review *i** concepts used in this model. Following [24], the QoS goals are expressed in terms of i* soft-goals and goals. The 15504 standard makes an explicit link between the purpose and the set of objectives to be fulfilled when executing BP that implement the service. So, as indicated in figure 4, purposes are modeled with a soft-goal and this soft-goal can be detailed by refining it into an equivalent collection of other soft-goals and/or goals associated with domain knowledge model. Because outcomes are objectively observable, they are modeled as goals (which can be further refined) and never with soft-goals. Indicators are added and modeled according their types, e.g. practices, work products and resources needed for the performance of the BP realizing the desired QoS. They are easily mapped into *i** concepts of task (for practices), *i** resources (for work products and resources) and actors (for resources).

To conclude, we would like to stress that our current contributions reported at the beginning of this section has convinced the ISO 15504 community that our GORE methodology was helpful in supporting the development of models compliant with ISO/IEC 15504. More details on this issue are reported in [30] and [24].

## 4 From Business to Software Services QoS Requirements

Our research centre is currently developing project management services dedicated to the network of SME partners, and aiming at defining and steering the projects the SME's partners of the Centre run. This system is a set of business services that support the management of projects where several partners are involved; it is realized by software components and human processes, that both cooperate to deliver the required business level objectives.

**Step 1: value analysis.** A value analysis, supported by e3-models, allowed us to identify that our research center could provide project management services to our SME network, and that we can therefore act as an actor in this network.

**Step 2: business service identification.** We identified our business services according to the normalized five successive phases of business collaboration of ISO Open EDI value-based model [25]: 1) planning, 2) identification, 3) negotiation, 4) actualization and 5) post-actualization. The business services required to support those phases are amongst others: "Define Proposal" in phase 1, "Steer Project" in phase 4, and "Negotiate Contract in phase 3". The business service "Define Proposal" allows a project manager to define and manage a project proposal involving multiple partners, including its review and acceptance.

**Step 3: strategic dependency model.** We adopted a goal-oriented technique to first identify the business level objectives of the business services. A strategic dependency model captures the relevant business services and their interactions with the business actors, as illustrated in figure 5. The objectives of the Project Owner (the SME) are supported by the objectives of the ProjectMgt Service Provider (our research centre), which is derived into three business services ("Define Proposal", "Steer Project", "Negotiate Contract"). The business services of Step 2 are therefore modeled by describing their objective in the strategic dependency diagram. The Project Partners (actor Partner) have a basic objective of participating to projects, which is not further detailed in this paper.

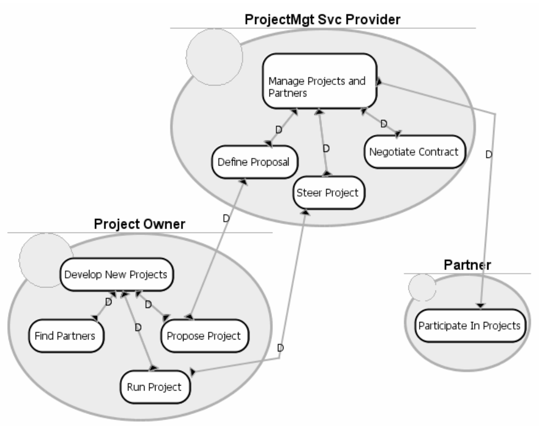
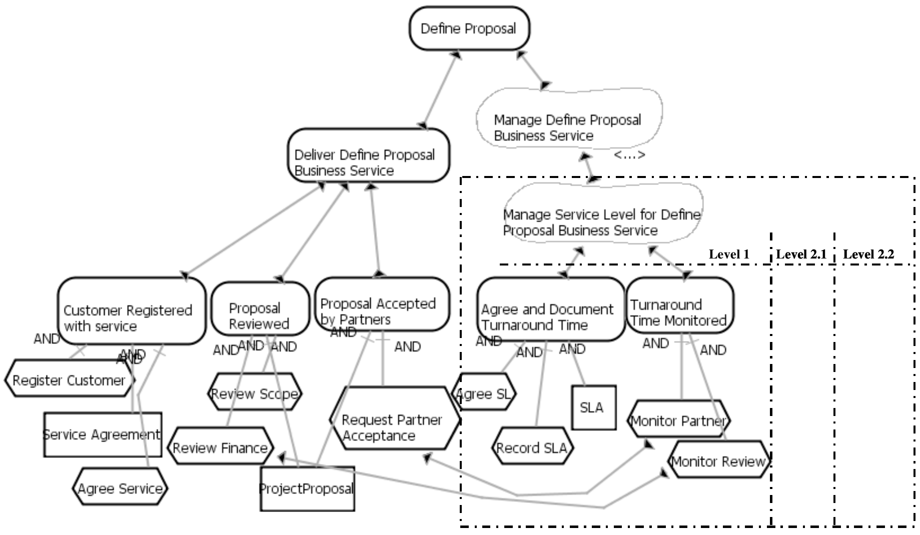


**Fig. 5.** Strategic Dependency Model – Context for Project Management Services

**Step 4: refining with service catalog.** Once identified and represented in their context, the business services objectives are refined by selecting and instantiating the appropriate requirements from the catalogue introduced in the previous section. In our case, regarding IT service management, and referring to figure 4, the collaboratively designed business service results in a goal tree, as illustrated in figure 6. In this excerpt of the model, level 1 of service level management has been selected, and instantiated to the “Define Proposal” business service. A specific service level requirement is shown in figure 6: the Turnaroud Time associated with the review and acceptance of the project proposal.



Define Proposal
Manage Define Proposal Business Service
<...>
Deliver Define Proposal Business Service
Manage Service Level for Define Proposal Business Service
Level 1
Level 2.1
Level 2.2
Customer Registered with service
Proposal Reviewed
Proposal Accepted by Partners
Agree and Document Turnaround Time
Turnaround Time Monitored
AND
Register Customer
Service Agreement
Agree Service
Review Scope
Review Finance
ProjectProposal
Request Partner Acceptance
Agree SLA
Record SLA
SLA
Monitor Partner
Monitor Review

**Fig. 6.** Business Requirements on Business Service “Define Proposal”

**Step 5: operationalization of core services.** Once specified from the business perspective, the business level objectives of the services are operationalised by allocating human and technical resources to perform the required activities. The business processes supporting the required tasks and work products are modeled, integrating the service level management tasks. The requirements of the “Define Proposal” business service are supported by two BP deployed by the service provider: a BP is dedicated to the registration of the customer with the service, and implements the core tasks “Register Customer” and “Agree Service”, but also the service level management tasks “Agree SLA” and “Record SLA”; a BP is dedicated to the actual service performance (and orchestrating the core business tasks). Figure 7 illustrates the BP model in a UML activity diagram.

**Step 6: identification and modeling of software services.** Although the business process “Register with Service” is not automated (the agreement on the service and on the required service level is still a human-based process), we decided to electronically record the SLA. The activity “Record SLA” of the business process becomes an automated task and requires the support of a software-based system. We come back to the intentional model (the goal tree in figure 8 that specifies the business service) to design this system: the requirement "Record SLA" is realised by a software-based task "Store eSLA". This task, together with other related tasks ("Retrieve eSLA", "Version eSLA") are tasks of the new software service "Manage eSLA". Some business level requirements are not realized with business processes, like "Monitor Review" and "Monitor Partners". We indeed decided to implement them directly with the supporting information system. They therefore are transformed into a software level objective ("Monitor the System"), which is the root objective of the software service "Monitoring Service".

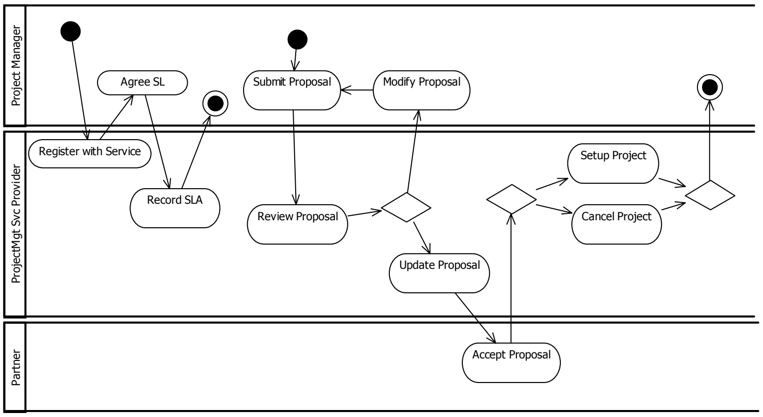


**Fig. 7.** Business Processes Supporting Business Service "Define Proposal"

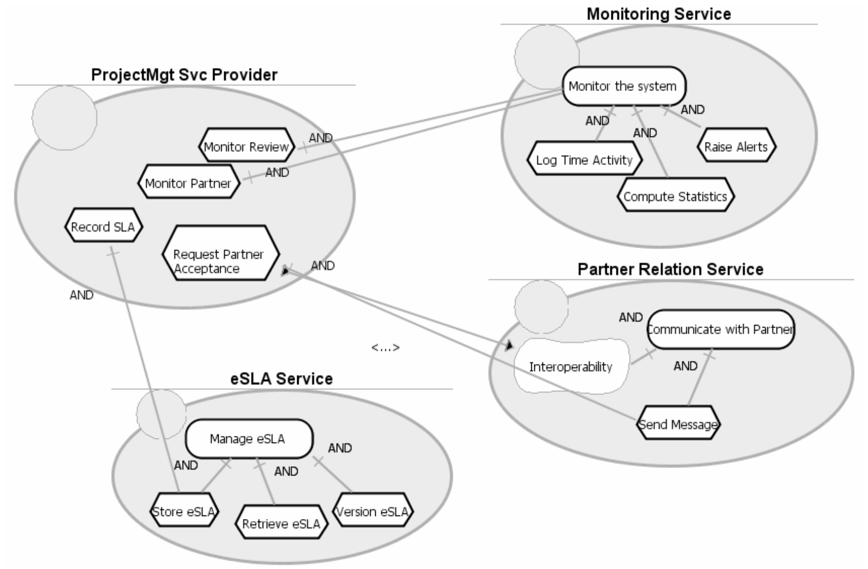


**Fig. 8.** Requirements for Software Services Supporting Business Level Requirements

**Step 7: management of services.** The introduction of the service management aspect as business level objectives introduces additional software services dedicated to the realization of the requirements associated with these business service management objectives. Figure 8 illustrates the requirements associated with these new software services, and shows an excerpt of the traceability we reach by using a goal-oriented modeling technique. The software services supporting the business level requirements are abstracted as new actors in the *i** model. We apply the same modeling steps that we used for modeling business service: the software service is modeled as a root goal (the objective of the service), which is refined into other soft goals, goals, tasks and resources. The software services are not only issued from the non-functional aspects (service management) associated with the business service. Figure 8 illustrates a software service identified from a functional requirement associated with the business service: "Request Partner Acceptance" justifies the introduction of a software service dedicated to the management of the partners ("Partner Relation Service"). The most prominent quality requirement associated with this service is interoperability, modeled as a soft goal of the software service.

## 5 Conclusion and Future Work

In this paper, we have reported on our framework related to the use of GORE techniques in the context of the management of QoS requirements expressed both at business and software levels. This framework applies to specific requirements whose are those inherent to norms, regulations and assurance domains. In these domains, information provided is often poorly structured and organized. In most cases, ambiguities, incompleteness, and even sometimes inconsistencies can be found in the available documents and sources of information. So, as it is claimed in e.g. [26, 27, 28], there is much sense to use requirements techniques (and GORE in particular) for the purpose of requirements clarification and formalization. With respect to these works, our work differs in its application to the characterization of e-services QoS as well as in the use of 15504, which allows us to organize requirements according to different capability levels that an organization may want to reach and expose to its customers. As illustrated in the presented case study, an organization can decide to adopt a service management of a level that can vary from 2 to 5. This variability issue is one that we need to further consider in the future. Analogously to [29], we need to consider variability associated goals graphs and requirements.

Another important issue considered in our approach is traceability. As explained and illustrated, a part of the QoS requirements at the software level can be systematically derived and traced to requirements identified at the business level. By establishing explicit traceability links between requirements at the two levels it is possible to demonstrate the compliance of software services with respect to regulations, norms and assurance recommendations. In a world where these compliance aspects are becoming crucial we feel that the proposed approach is a very first answer in the services domain. As part of our future work, like in [26] we intend to better formalize the traceability model underlying our framework in order to support a more effective deployment. We also intend to further refine our approach through the handling of new real business cases, which require this business/IT services alignment perspective.

## References

1. Zarvić, N., Wieringa, R.J., van Daneva, M.: Towards Information Systems Design for Value Webs. In: Proceedings of Workshops of CAiSE 2007, Trondheim, Norway, pp. 453–460. Tapir Academic Press, London (2007)
2. ISO, ISO/IEC 9126-1: Software Engineering – Product Quality – Part 1: Quality Model (2001)
3. Gordijn, J., Akkermans, H.: Value based requirements engineering: Exploring innovative e-commerce idea. Requirements Engineering Journal 8(2), 114–134 (2003)
4. Parasuraman, A., Zeithaml, V.A., Berry, L.: A Conceptual Model of Service Quality and Its Implications for Future Research. Journal of Marketing 49, 41–50 (1985)
5. O'Sullivan, J., Edmond, D., ter Hofstede, A.: What's in a Service? Towards Accurate Descriptions of Non-Functional Service Properties. Distributed and Parallel Databases 12, 117–133 (2002)
6. IT Infrastructure Library – Service Delivery, The Stationery Office Edition (2000); ISBN 011 3308671
7. ISO, ISO/IEC 20000-1:Information Technology – Service Management – Part 1: Specification (2005)
8. ISO, ISO/IEC 27005: Information Technology – Security Techniques – Information Security Risk Management (2008)
9. The Sarbanes-Oxley Act of 2002, Pub. L. No. 107-204, 116 Stat. 745, USA. Public Company Accounting Reform and Investor Protection Act (SOX) (July 30, 2002)
10. COSO (1994) Internal Control – Integrated Framework, CSOTC, USA, (Retrieved December 1, 2007), `http://www.coso.org/`
11. Basel Committee on Banking Supervision: International Convergence of Capital Measurement and Capital Standards. Bank for International Settlements Press & communication, Basel (2004)
12. ISO, ISO/IEC 15504: Information Technology – Process Assessment: Part1 - Part5 (2003)
13. Yu, E., Mylopoulos, J.: Understanding "Why" in Software Process Modelling, Analysis, and Design. In: Proceedings of 16th International Conference on Software Engineering (1994)
14. `http://www.gestiondeprojet.lu`
15. Henderson, J.C., Venkatraman, N.: Strategic Alignment: Leveraging Information Technology for Transforming Organizations. IBM Systems Journal 38, 472–484 (2004)
16. Dardenne, A., van Lamsweerde, A., Fickas, S.: Goal Directed Requirements Acquisition. Science of Computer Programming 20, 3–50 (1993)
17. Thevenet, L.H., Salinesi, C.: Aligning IS to Organization's Strategy: The InStAlMethod. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) CAiSE 2007 and WES 2007. LNCS, vol. 4495, pp. 203–217. Springer, Heidelberg (2007)
18. Barafort, B., Humbert, J.-P., Poggi, S.: Information Security Management and ISO/IEC 15504: the link opportunity between Security and Quality. In: Proc. Conf. SPICE 2006, Luxembourg (2006)
19. Valoggia, P., Di Renzo, B.: Assessment and Improvement of Firm's Knowledge Management Capabilities by Using a KM Process Assessment Compliant to ISO/IEC 15504. A Case Study. In: Proc. Conf. SPICE 2007, Seoul, Korea (2007)
20. Rifaut, A., Di Renzo, B., Picard, B.,, M.: ISO/IEC 15504, a Basis for Generally Accepted Sound Process Models in Financial Institutions: A Case Study about Venture Capital Fund Management. In: Proc. Conf. SPICE 2008, Nuremberg (2008)
21. Rifaut, A., Picard, M., Di Renzo, B.: ISO/IEC 15504 Process Improvement to Support Basel II Compliance of Operational Risk Management in Financial Institutions. In: Proc. Conf. SPICE 2006, Luxembourg (2006)

22. Barafort, B., Renault, A., Picard, M., Cortina, S.: A Transformation process for Building PRMs and PAMs Based on a Collection of Requirements – Example with ISO/IEC 20000. In: SPICE 2008, Nuremberg, Germany (2008)
23. CMM® (2007): Capability Maturity Model for Software, Software Engineering Measurement and Analysis Initiative, Carnegie Mellon University, USA (Retrieved December 1, 2007), http://www.sei.cmu.edu/cmm
24. Rifaut, A., Dubois, E.: Using Goal-Oriented Requirements Engineering for Improving the Quality of ISO/IEC 15504 based Compliance Assessment Frameworks. In: Proc. IEEE Intl. Conf. On Requirements Engineering (RE 2008), Barcelona. IEEE Press, Los Alamitos (2008)
25. ISO, ISO/IEC 14662: Information Technology – Open EDI Reference Model (2004)
26. Breaux, T.D., Vail, M., Anton, A.: Towards Regulatory Compliance: Extracting Rights and Obligations to Align Requirements with Regulations. In: Proc. 14th IEEE International Requirements Engineering Conference (RE 2006), pp. 46–55. IEEE Press, Los Alamitos (2006)
27. Ghanavati, S., Amyot, D., Peyton, L.: Towards a Framework for Tracking Legal Compliance in Healthcare. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) CAiSE 2007 and WES 2007. LNCS, vol. 4495, pp. 218–232. Springer, Heidelberg (2007)
28. Karagianis, D., Mylopoulos, J., Schwab, M.: Business Process-Based Regulation Compliance: The Case of the Sarbanes-Oxley Act. In: Proc. 15th IEEE International Requirements Engineering Conference (RE 2007), pp. 315–321. IEEE Press, Los Alamitos (2007)
29. Lapouchnian, A., Yu, Y., Mylopoulos, J.: Requirements-Driven Design and Configuration Management of Business Processes. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 246–261. Springer, Heidelberg (2007)
30. Rifaut, A.: Goal-Driven Requirements Engineering for Supporting the ISO 15504 Assessment Process. In: Richardson, I., Abrahamsson, P., Messnarz, R. (eds.) EuroSPI 2005. LNCS, vol. 3792, pp. 151–162. Springer, Heidelberg (2005)

# Active Energy-Aware Management of Business-Process Based Applications
## Position Paper

Danilo Ardagna, Cinzia Cappiello, Marco Lovera, Barbara Pernici, and Mara Tanelli

Politecnico di Milano, Dipartimento di Elettronica e Informazione, Italy
lastname@elet.polimi.it

**Abstract.** Energy management is becoming a priority in the design and operation of complex service-based information systems, as the energy costs of IT infrastructures increase. This paper aims at introducing a novel interdisciplinary approach for the development of advanced active energy-aware business process applications, based on expertise from several research areas: Web service technologies, data deduplication, optimization, performance evaluation, model identification, robust and predictive control. The basic idea is that enforcing energy efficiency goals for the development of green business process systems can only be achieved by recognizing their multi-layer feedback nature, which can be successfully exploited by combining IT methodologies with methods and tools from systems and control theory.

**Keywords:** Green IT, Business Process Optimization, Resource Allocation, QoS management, SLA, System identification and Control Theory.

## 1 Introduction

Energy management is rapidly becoming a priority in the design and operation of complex business process-based systems, as the impact of energy consumption associated with IT infrastructures increases [4]. The growth in the number of servers and the increasing complexity of the network infrastructure have caused an enormous spike in electricity usage. Power consumption per rack has increased from 1kW in 2000 to 8kW in 2006 and is expected to rise to 20kW by 2010. IT analysts predict that, by 2012, up to 40% of IT budget will be consumed by energy costs [10]. This trend is striving green computing activities in the industry research agenda (see for example IBM's project *Big Green* [1] and HP's *Green up* initiative [2]).

Research and innovation within industry offer the possibility for significant improvement in energy efficiency management for computing systems. For example, software as a service suggests economies of scale. In recent years, large

[1] http://www-03.ibm.com/press/us/en/photo/21514.wss
[2] http://www.hp.com/hpinfo/newsroom/feature-stories/2007/07-360-greenup.html

© Springer-Verlag Berlin Heidelberg 2008

service centers have been setup to provide computational capacity on demand to many customers who share a pool of IT resources.

In the context of Web services and Service Oriented Architecture (SOA) based systems, service centers need to comply also to the Service Level Agreements (SLAs) stipulated with their customers. At run time, service requestors address their invocation to the most suitable provider according to their Quality of Service (QoS) preferences. QoS requirements are difficult to satisfy because of the high variability of Internet workloads. It is difficult to estimate workload requirements in advance, as they may vary by several orders of magnitude within the same business day [15]. To handle workload variations, many service centers employ autonomic self-managing techniques (see, e.g., [3,16]), such that resources are dynamically allocated among running Web services based on short-term demand estimates. The goal is to meet the application requirements while adapting the IT architecture.

Current work in sustainable and energy aware computing suggests to provide services with a trade-off between performance and energy consumption. There exists a significant body of resource management work focused on attaining maximum performance (see, e.g., [16,11]). It is important that energy efficiency be given a role of equal importance in resource management. Service centers have mainly focused their early energy efficiency efforts on consolidating and virtualizing servers. Yet, in spite of significant gains in server consolidation and energy savings, storage remains a gaping hole in the enterprise service center. IDC predicts storage annual growth around 60% due to increased business continuity requirements which necessitate the replication of data. As a result, storage consumes more and more power, and the same principles that govern server energy savings should be applied to the storage sub-system as well [7].

The aim of our research is to develop novel energy-aware resource allocation mechanisms and policies for SOA, and business process-based applications via an interdisciplinary approach. The main goal is to provide services with QoS guarantees, while minimizing the energy consumption of the computing infrastructure.

The remainder of the paper is organized as follows. Section 2 illustrates our energy aware business process framework. Preliminary results that have been already achieved at the lower layers of the framework are reported in Section 3. A summary of the most relevant literature proposals is reported in Section 4. Conclusions are finally drawn in Section 5.

## 2 Active Energy Aware Resource Management Framework

In modern service based systems, workload variations and SLA management of business processes lead to the problem of efficient use of resources and the reduction of energy expenses. Energy management at a service center is a complex problem and, in our vision, can be achieved by considering three distinct layers (see Figure 1): the *Process layer*, the *Infrastructure layer*, and the *Control layer*. Each layer will be thoroughly described in the following Sections.

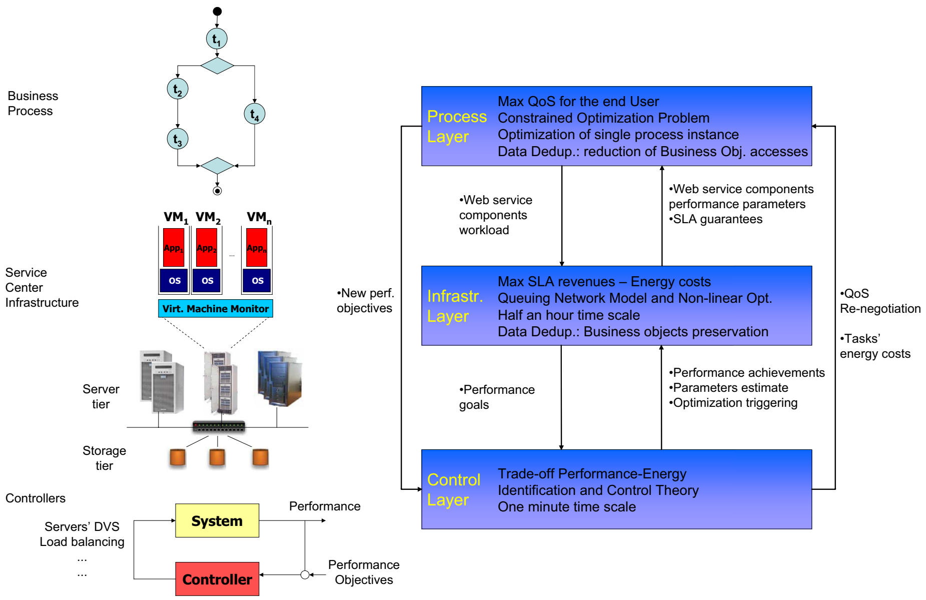


**Fig. 1.** Active Energy Aware Resource Management Framework

### 2.1 Process Layer

The Process layer manages business process end user applications. In advanced SOA systems, complex applications are described as abstract business processes (composed by tasks $t_1$, $t_2$,...$t_N$ in Figure 1), which are executed by invoking a number of available Web services. End users can specify different preferences and constraints and service selection can be performed by dynamically identifying the best set of services available at run time. Indeed, a service provider can offer functionally equivalent business processes and/or Web service components with several QoS profiles characterized by different energy costs. At this layer, resource allocation techniques provided by the literature [18] focused on the maximization of the QoS for the end user, since this has an impact on user's satisfaction, provider's reputation, and, hence, on provider's revenue in the long term. Web service selection results in an optimization problem (whose goal is to optimize a single process instance), which has been studied both in the research areas of service oriented computing for business processes and of grid environments [19]. Anyway, performance issues are usually not considered and energy consumption has always been neglected. Several research contributions [1,19] have formalized the resource allocation problem as a constrained optimization problem. In such an optimization process, the performance of Web service components have to be guaranteed by the low level layers of the infrastructure. Furthermore, QoS optimization does not analyze the process efficiency in terms of accesses and management of business objects. Data redundancy, for example, affects energy consumption since multiple copies of the same data have to be kept synchronized. At this level, data deduplication techniques [6] can be applied in order to identify and merge different copies of the same object and thus reduce the number of accesses to business objects. Green IT calls for a new approach to data management in business processes, since at design time the trade-off between energy consumption and the availability provided by data replication should be evaluated carefully.

In [1], we have proposed an optimization technique for QoS maximization based on mixed integer linear programming which has been demonstrated to be efficient under stringent constraints and for large processes instances. In current work we are extending that approach in order to include explicitly energy issues and object replica management in the QoS evaluation.

### 2.2 Infrastructure Layer

The Infrastructure layer focuses on workload variations and on the trade-off between the performance of Web service components (which have to be guaranteed to the first layer), and energy consumption. At this layer, Web service components invoked by business processes are mapped to multi-tier server applications which are currently executed by independent Virtual Machines (VMs) in hosted virtualized environments (e.g., based on VMWare or Xen). Physical resources (e.g., CPU, disks, communication network) are partitioned into multiple virtual ones, creating isolated virtual machines each running at a fraction of the physical system capacity. Each VM is usually dedicated to serve a single application. Autonomic self-managing techniques are currently implemented by *network controllers* which can establish the set of applications (VMs) executed by each server (*VM placement* sub-problem), the request volumes at various servers (*load balancing* sub-problem), and the capacity devoted to the execution of each application (VM) at each server (*capacity allocation* sub-problem). The network controller can also decide to turn on or off servers depending on the system load or to reduce the frequency of operation of servers [9] (*servers provisioning* and *frequency scaling* sub-problems). Overall, autonomic self-managing techniques are designed to maximize the SLA profits (including revenues and penalties), while balancing the cost of using resources (including energy and air conditioning) [11,15,16]. In [2] we have designed resource allocation techniques for the management of multi-tier virtualized systems. Allocation policies provide a joint solution to the server provisioning, frequency scaling, VMs placement, load balancing and capacity allocations problems. The joint problem has been formalized as a mixed integer non linear programming problem whose goal is to maximize the profits associated with multiple class SLAs while minimizing energy cost. The problem is NP-hard and the inclusion of energy costs in the objective function keeps its soloution very challenging. We developed heuristic solutions by implementing a local-search algorithm which can solve problem instances up to 400 physical servers and 100 requests classes within a half an hour optimization time constraint.

Energy efficiency in storage can be achieved by adopting data deduplication also at this layer. Indeed, the basic idea is to store only data changes on storage devices, while redundant data is replaced with a pointer to the unique data copy. In our approach data deduplication is also applied for archiving purposes focusing on high level application requirements. For example, business objects can undergo accidental or intentional deletions due to storage limitation or to under-evaluation of their importance. This loss of information can be avoided by detecting and preserving important objects. We have started improving data quality techniques for the identification of the only relevant copy to be preserved.

## 2.3 Control Layer

The differentiation between the Infrastructure and the Control layers is mainly motivated by the different time scales corresponding to the respective actions: server provisioning and VM placement decisions, i.e., the activities corresponding to the Infrastructure, are taken about every half an hour [3,11,16] because they introduce a significant system overhead. At this layer, the system is usually described by means of a performance model based on queueing theory embedded within an optimization framework. The resource allocation problem can be formalized as a non linear and mixed integer optimization problem which has to be solved within strict time constraints.

Load balancing, capacity allocation, and frequency scaling problems, on the other hand, have tight time constraints but imply a relatively low computation burden so they can be actuated in the system (without introducing any overhead) every few minutes. The Control layer is therefore associated with such operations. The main problem here is given by the fact that queueing theory is based on the assumption that the overall system is at a *steady state* and therefore cannot accurately model system transients. Hence, models based on queueing theory are effective for open-loop planning on a medium term time horizon (i.e., half an hour, [3,12]) but represent very crude approximations of the actual behavior of the system during transients. In particular, relying on such models for closed-loop control system design, as is frequently done in the literature [3,16], can only lead to poor performance in terms of QoS and, as a consequence, to highly inefficient systems in terms of energy consumption. These considerations lead to a different view of the third layer of the framework which aims at tackling workload variations and adjusting the system configuration within a very short time frame (e.g., every minute). We argue that these goals can only be attained by using dynamical models which can accurately represent system transients under varying workload conditions and by relying on genuine control-theoretic approaches for the design. The third layer is therefore viewed as a feedback loop, where the SLA objectives (determined at the second layer) are translated into set-points for the response time of the servers (possibly different according to the customer classes) and tracking performance is traded-off with energy savings.

In [14] we identified a control-oriented dynamic model of an application server based on the Linear Parameter Varying (LPV) framework capable of capturing system behavior at a very fine-grained time resolution (e.g., minutes or seconds), with an accuracy suitable for control purposes. This identification process is the first step in order to design a closed-loop controller for service center infrastructures able to meet SLAs requirements while minimizing energy costs.

### 2.4 Need for an Integrated Approach

The solutions for resource management provided in the literature usually consider each of the above described layers in isolation.

We argue that, in order to achieve energy efficiency in modern service centers, an integrated approach is needed. Indeed, the solutions at different layers have important interrelations.

For example, the solution of the optimization problem at the Process layer determines the request volume of the server applications and VMs of the Infrastructure layer. At the Process layer, we advocate a totally new approach in the design of process based applications. Business process designers have to consider that each operation in the service center has a cost and energy efficiency can be achieved only by considering users' behaviour and concurrent execution of multiple process instances. Users' behaviour has an impact on system workload. As it will be discussed also in the next Section, it is more difficult to manage system resources under peak workload and delaying the execution of some tasks could reduce the power budget of the service center and increase its energy efficiency [4]. Similarly, with the current practice of server and storage consolidation, multiple process instances compete in the access to shared resources at the Infrastructure layer. This implies an additional coupling between process and infrastructure, whose management calls for additional efforts in order to optimize the execution of multiple instances and include energy issues in the business process QoS evaluation.

The interrelations among the models adopted at the Infrastructure and Control layers are also very promising. Recently [20], control techniques have been applied for the estimate of the performance model parameters which govern the optimization decisions at the Infrastructure layer. The analysis of the different time scales which govern the control decisions at the second and third layers has not yet been addressed in the literature and constitutes a challenging and important research problem. Nowadays, control time horizons are determined by technology constraints (e.g., the time and overhead required to move a VM). The Control layer could also govern and adapt the control time horizon used at the Infrastructure layer for example by triggering the global optimization only when needed instead of periodically as in the current practice. Furthermore, from a theoretical point of view, system stability has been demonstrated only for the local controllers which work at the Control layer, but the stability of the global decisions taken at the Infrastructure layer has not yet been proven.

Finally, there exist interrelations also between the Control and Process layers. Indeed, the Control layer is based on a very fine time grain, it can describe accurately the system behavior both under transient and stationary conditions, and hence can determine accurately the energy consumption associated with each task operation. Associating energy costs with task operation is important in order to evaluate precisely business process *green* key performance indicators [4].

Furthermore, if performance objectives can not be fulfilled at the Process layer, the QoS for the end user should be re-negotiated as the profile of Web service components. The implementation of the overall framework is part of our ongoing work. Our aim is the development of an integrated energy-aware middleware for execution of green business processes with QoS guarantees. The next Section is devoted to the presentation of the preliminary results which have been already achieved at the Infrastructure and Control layers.

## 3 Experimental Results

### 3.1 Infrastructure Layer Preliminary Results

In this Section, we compare the results which can be achieved by our energy-aware resource allocation policies [2] with respect to the top performing state-of-the art solutions currently implemented in real systems (IBM Websphere and Tivoli). In particular our solution is compared with

– the server provisioning and VM placement solutions proposed in [15];
– the load balancing and capacity allocation solutions proposed in [11,16].

In the following we will refer to Pacifici et al. works [11,15,16] also as the *alternative solution.*

The comparison is based on realistic workloads created from a trace of requests obtained by the Web site of a large University in Italy. The system includes almost 100 servers and the trace contains the number of request sessions, on a per-hour basis, over a one year period (01/01/06 to 31/12/06). Realistic workloads are built by assuming request arrivals to follow non-homogeneous Poisson processes with rates changing every hour according to the trace. Several analyses of actual e-commerce site traces, see for example [17], have shown that the Internet workload follows a Poisson distribution as first approximation. From this logs, we have extracted 10 requests classes which correspond to the days with the highest workloads experienced during the year (see Figure 2).

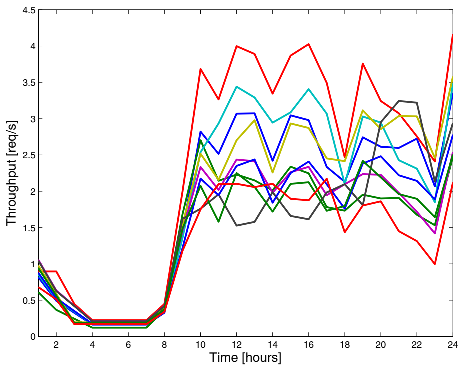


**Fig. 2.** Requests throughput

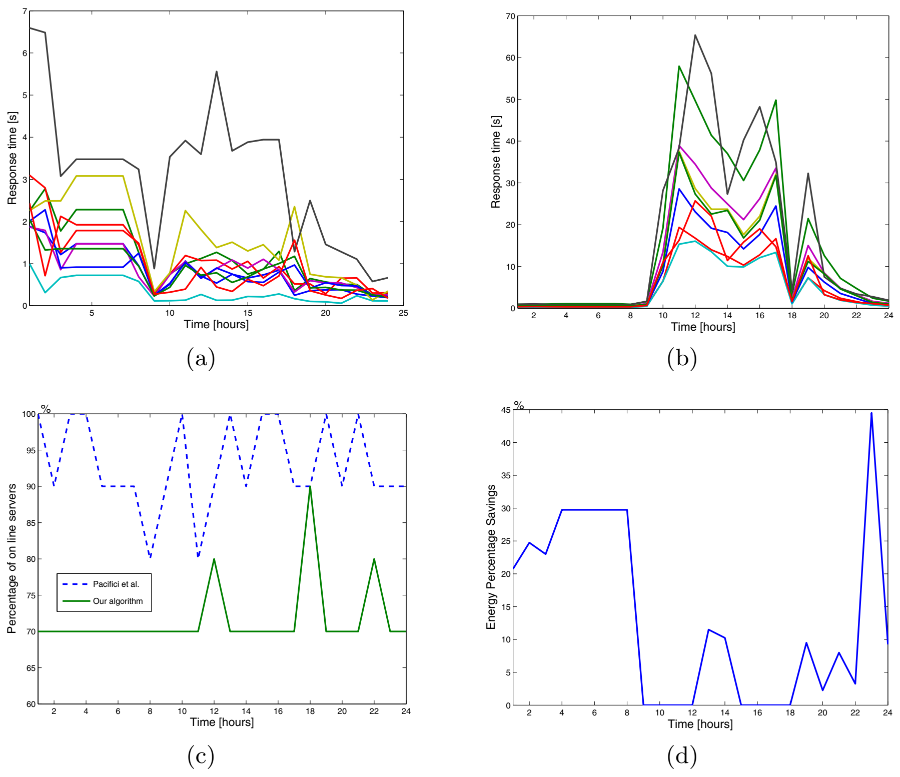


**Fig. 3.** (a) Response times obtained by our algorithm; (b) Response times obtained by the alternative solution; (c) Percentage of on line servers in Pacifici et al. comparison; (d) Percentage energy saving with respect to static allocation

The comparison is based on simulation tests where long-tails distribution for service times have been considered (in particular, log-normal distributions as observed for several real applications [17]). Results have been obtained with the Anylogic 6.0 simulator and show that our algorithm performs much better during the peak hours (between 9.00-20.00) while, under light load (1.00-8.00, 21.00-24.00), the two solutions provide similar results. Our solution (see plots in Figures 3(a) and 3(b)) provides response times one order of magnitude better than the alternative one. This result is achieved since the alternative solution always evenly balance the workload among running servers, while our algorithm mainly assigns dedicated servers to VMs and that provides better performance (see [2]). The plot reported in Figure 3(c) shows the percentage of servers available at the service center adopted by the two solutions. Our solution obtains a more efficient use of the resources since, excluding the initial part of the day, it is able to provide better response times while adopting a lower number of servers. Overall, during the 24 hours, our net revenues, which include SLA profits and energy costs, are about 30% higher than the alternative ones.

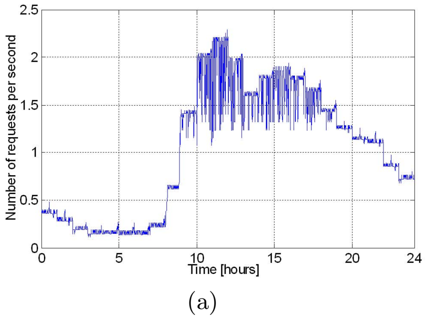
Finally, in order to evaluate the energy savings which can be achieved by a static allocation, we considered another scenario where: (i) the service center (which includes 400 servers and 100 requests on four tiers) always run at full capacity, (ii) only energy cost are considered, and (iii) SLA revenues are neglected. Results are reported in Figure 3(d). Significant energy savings can be obtained under light load conditions. By varying system parameters (e.g., energy costs, air conditioning costs, etc.) over the 24 hours the energy savings range between 13 and 25%.

## 3.2 Control Layer Preliminary Results

As mentioned in Section 2.3, the main issue associated with the design of the Control layer is the development of dynamical models of the Infrastructure layer servers capable of capturing transients. In fact, our aim is to obtain a control-oriented dynamical description of the servers' behavior. The Control layer will need to operate at a very fine grained time resolution (e.g., seconds), in order to ensure that SLAs requirements are met while minimizing energy costs. As the behavior of the server response time is highly time varying and the workload conditions change substantially within the same business day, a modelling class capable of capturing such effects has to be considered.

To this purpose, Linear Parametrically Varying (LPV) models, which are linear models whose dynamics are scheduled by external time-varying parameters, seem very promising for modeling such systems as they can relate the relevant control input (such as, e.g., the server operating frequency) to the output (i.e., the response time of the individual server), taking into account the role of the sustained server workload (as measured by, e.g., the requests arrival rate $\lambda$ and the requests service time $s$) which enters the model as a scheduling parameter. It is apparent from the recent literature (see, e.g., [12] and the references therein) that black-box identification methods represent a very promising approach to this modelling problem. In this Section we summarize the preliminary results which have been achieved in the identification of state-space LPV models for a single class, single tier Web service application (see [14] for further details) whose goal is to determine a model description able to capture Web service dynamics. We assume that the physical servers available at the service center implements the Dynamic Voltage Scaling (DVS) mechanism which varies both CPU supply voltage and operating frequency. The adoption of DVS is very promising, as power consumption is proportional to the cube of the operating frequency, while servers performance varies linearly with the operating frequency [2]. Furthermore, DVS, unlike hibernating and restoring, does not introduce any system overhead.

In the experimental framework, a workload generator and a micro-benchmarking Web service application have been used. The workload generator is based on a custom extension of the Apache *JMeter* 2.3.1 workload injector, which allows to generate workload according to an open model [8] with a Poisson arrival process. The Web service is a Java servlet designed to consume a fixed amount of CPU time generated according to deterministic (for identification purposes),



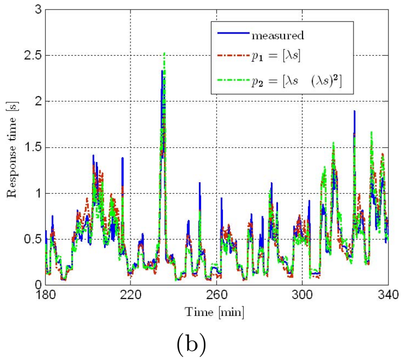


**Fig. 4.** (a) Time history of the request rate applied during a validation test; (b) Detail of the measured (solid line) and the response time obtained with $\Delta t = 10\,\mathrm{s}$ an LPV model with $p_1 = \lambda\, s$ (dashed line) and $p_2 = [\lambda\, s\ (\lambda\, s)^2]$ (dash-dotted line) on identification data in with $q = 4$

Poisson, Pareto and log-normal distributions (for validation). The adoption of a micro benchmarking application allows the validation of the effectiveness of our approach both for workload intensive and for computationally intensive applications. Furthermore, the CPU time standard deviation of the micro benchmarking application has been varied in order to verify if LPV models performance depends on the variability of the CPU time distribution: the standard deviation $\sigma[s]$ has been chosen as $q$ times the average of the service time distribution $E[s]$, i.e., $\sigma[s] = q\,E[s]$, where $q$ was set equal to 2, 4 and 6. For model validation, the incoming workload reproduces the 24 hour trace corresponding to the peak workload considered in Section 3.1 where a Gaussian noise (see Figure 4(a)) proportional to the workload intensity has been added as in [9]. To quantitatively evaluate the models, two metrics have been considered: the percentage Variance Accounted For (VAF), defined as $VAF = \left(1 - \frac{Var[y_k - \widehat{y}_k(\theta)]}{Var[y(k)]}\right)$, where $y_k$ is the measured signal (i.e., application response time), and $\widehat{y}_k(\theta)$ is the output obtained from the simulation of the identified model, and the percentage average simulation error $e_{avg}$, computed as $e_{avg} = \left(\frac{E[|y_k - \widehat{y}_k(\theta)|]}{E[|y_k|]}\right)$.

In the LPV identification, we analysed two possible choices for the scheduling parameters, namely the server utilization (i.e., $p_1 = \lambda\, s$, see also [12]) and the server utilization and its square (that is, $p_2 = \left[\lambda\, s\ (\lambda\, s)^2\right]$). The system output is the service response time.

The identification data were processed to extract the average values over a sampling interval $\Delta t = 10\mathrm{s}$ and two LPV second order models, one with $p_1 = \lambda\, s$ and the other with $p_2 = [\lambda\, s \quad (\lambda\, s)^2]$ have been identified (see Figure 4(b) for a plot of a detail of the results).

As can be seen from the Figure, the models are capable of providing a response time which correctly follows the peaks of the measured one. Results reported in

**Table 1.** Performance of the identified models with $\Delta t = 10$s on validation data

| Valid. Performance - LPV | q=2 | | q=4 | | q=6 | |
|---|---|---|---|---|---|---|
| $\Delta$t = 10 s | $(p_1)$ | $(p_2)$ | $(p_1)$ | $(p_2)$ | $(p_1)$ | $(p_2)$ |
| VAF on 24h | 58.31% | 74.14% | 54.01% | 71.50% | 58.85% | 74.52% |
| VAF light load (1-8, 21-24)h | 86.68% | 91.58% | 78.60% | 80.20% | 76.57% | 73.61% |
| VAF heavy load (9-20)h | 54.34% | 71.59% | 48.50% | 67.10% | 57.15% | 77.52% |
| $e_{avg}$ on 24h | 25.70% | 18.36% | 20.30% | 7.40% | 31.87% | 31.67% |
| $e_{avg}$ light load (1-8, 21-24)h | 28.78% | 12.55% | 20.02% | 2.50% | 44.47% | 41.35% |
| $e_{avg}$ heavy load (9-20)h | 26.22% | 21.19% | 22.50% | 9.25% | 28.88% | 29.05% |

Table 1 also show that the performance of LPV models are almost independent on the value of $q$, i.e., the models are robust to the variability of the service time distribution of the Web service application.

## 4 Related Work

The most relevant contributions provided in the literature [1,13,16,19] usually consider each sub-problem addressed at different layers of our framework in isolation. Business process optimization has been applied in context-aware business processes and e-science research fields. The literature has provided *three generations* of solutions [1]. First generation solutions implemented *local* approaches [18,19] which select component Web services one at the time by associating the running task to the best candidate service which supports its execution. Local approaches can guarantee only local QoS constraints, i.e., candidate services are selected according to a desired characteristic, e.g., the price of *a single Web service* is lower than a given threshold. Second generation solutions proposed *global* approaches [5,18,19]. The set of services which satisfy the process constraints and user preferences for the whole application are identified before executing the process. In this way, QoS constraints can predicate at a global level, i.e., constraints posing restrictions over the *whole composed service execution* can be introduced. Finally, third generation techniques [1] focus on the execution of processes under severe QoS constraints.

The autonomic management of the infrastructure layer has been largely considered by the research community and some features are currently implemented in commercial products, e.g., IBM Tivoli [11,15,16]. Early solutions switched servers on and off [3], while more recent proposals [9] have started reducing the frequency of operation of servers by exploiting the DVS mechanisms implemented in new servers.

Finally at the control layer, modern approaches provide solution for the QoS management of services infrastructures. LPV models have been adopted in Qin and Wang work [12] in order to implement an autonomic controller able to provide performance guarantees by means of DVS.

## 5 Conclusions

Climate debate and sustainable growth concern over energy use will strive green computing in the Service area research agenda [4]. In our work, we have provided solutions able to determine QoS and energy trade-off at the individual layers of our framework. Ongoing work is focusing on the analysis of the different time scales and the interrelations which characterize the resource managers working at the different layers. The aim is to exploit information from the lower layers to quantitatively estimate the energy consumption required for business processes and component Web services execution.

## Acknowledgments

This work has been partially supported by the project Q-ImPrESS and S-Cube NoE funded under the European Union's Seventh Framework Programme (FP7).

## References

1. Ardagna, D., Pernici, B.: Adaptive Service Composition in Flexible Processes. IEEE Transactions on Software Engineering 33(6), 369–384 (2007)
2. Ardagna, D., Trubian, M., Zhang, L.: Energy-Aware Autonomic Resource Allocation in Multi-tier Virtualized Environments. Politecnico di Milano, Dipartimento di Elettronica e Informazione Technical report number 2008. 13 (July 2008)
3. Ardagna, D., Trubian, M., Zhang, L.: SLA based resource allocation policies in autonomic environments. Journal of Parallel and Distributed Computing 67(3), 259–270 (2007)
4. Barroso, L.A., Hoolzle, U.: The case for energy-proportional computing. IEEE Computer 40 (2007)
5. Canfora, G., Penta, M., Esposito, R., Villani, M.L.: QoS-Aware Replanning of Composite Web Services. In: ICWS 2005 Proc., Orlando (2005)
6. Elmagarmid, A.K., Ipeirotis, P.G., Verykios, V.S.: Duplicate record detection: A survey. IEEE Transactions on Knowledge and Data Engineering 19(1), 1–16 (2007)
7. Karlsson, M., Karamanolis, C., Zhu, X.: Triage: Performance Diff. for Storage Systems Using Adaptive Control. ACM Transactions on Storage 1(4), 457–480 (2005)
8. Kleinrock, L.: Queueing Systems. John Wiley and Sons, Chichester (1975)
9. Kusic, D., Kandasamy, N.: Risk-Aware Limited Lookahead Control for Dynamic Resource Provisioning in Enterprise Computing Systems. In: ICSOC Proc. (2006)
10. Metha, V.: A Holistic Solution to the IT Energy Crisis (2007)
11. Pacifici, G., Spreitzer, M., Tantawi, A.N., Youssef, A.: Performance Management for Cluster-Based Web Services. IEEE Journal on Selected Areas in Communications 23(12) (December 2005)
12. Qin, W., Wang, Q.: Modeling and control design for performance management of web servers via an LPV approach. IEEE Transactions on Control Systems Technology 15(2), 259–275 (2007)
13. Rolia, J., Cherkasova, L., McCarthy, C.: Configuring Workload Manager Control Parameters for Resource Pools. In: IEEE NOMS, Vancouver, Canada (April 2006)

14. Tanelli, M., Ardagna, D., Lovera, M.: LPV model identification for power management of web services. In: IEEE Multi-conference on Systems and Control (2008)
15. Tang, C., Steinder, M., Spreitzer, M., Pacifici, G.: A scalable application placement controller for enterprise data centers. In: WWW 2007 (2007)
16. Urgaonkar, B., Pacifici, G., Shenoy, P.J., Spreitzer, M., Tantawi, A.N.: Analytic modeling of multitier Internet applications. ACM Transactions on the Web 1(1) (2007)
17. Williams, A., Arlitt, M., Williamson, C., Barker, K.: Web Workload Characterization: Ten Years Later. In: Web Content Delivery. Springer, Heidelberg (2005)
18. Yu, T., Zhang, Y., Lin, K.-J.: Efficient algorithms for web services selection with end-to-end qos constraints. ACM Transactions on the Web 1(1), 1–26 (2007)
19. Zeng, L., Benatallah, B., Dumas, M., Kalagnamam, J., Chang, H.: QoS-aware middleware for web services composition. IEEE Transactions on Software Engineering 30(5) (May 2004)
20. Zheng, T., Woodside, C.M., Litoiu, M.: Performance model estimation and tracking using optimal filters. IEEE Transactions on Software Engineering 34(3), 391–406 (2008)

# An Architecture for Managing the Lifecycle of Business Goals for Partners in a Service Network*

Marina Bitsaki[1], Ohla Danylevych[2], Willem-Jan van den Heuvel[3], George Koutras[1], Frank Leymann[2], Michele Mancioppi[3], Christos Nikolaou[1], and Mike Papazoglou[3]

[1] Computer Science Department, Universuty of Crete, Greece
{bitsaki,koutras,nikolau}@tsl.gr
[2] Institute of Architecture of Application Systems, University of Stuttgart, Germany
{olha.danylevych,frank.leymann}@iaas.uni-stuttgart.de
[3] Department of Information Systems and Management, Tilburg University, Netherlands
{W.J.A.M.vdnHeuvel,michele.mancioppi,mikep}@uvt.nl

**Abstract.** Networks of interdependent organizations cooperate to produce goods or, nowadays, services that are of value to their markets as well as to the participating organizations. Such co-operations can be supported by corresponding business processes which are based on SOA technology. Developing and managing SOA-based business processes in such service networks necessitates a *comprehensive* architecture which is on the one hand grounded on solid design principles, and on the other hand capturing best-practices and experiences. Such an architecture is currently lacking. This paper outlines a first attempt to develop and validate an architecture for developing, monitoring, measuring and optimizing SOA-enabled business processes in service networks. A case study from the telecommunications industry is analyzed, and different aspects of service networks are addressed.

**Keywords:** Service Value Network, Key Performance Indicator, Business Process Management, Business Activity Monitoring.

## 1 Introduction

The emerging service economy and the advances in information technology have dramatically increased the complexity of understanding how organizations evolve within a world of interactions and partnerships. Instead of large, vertically integrated organizations, we observe the emergence of globe-spanning networks of interdependent companies that cooperate to provide value to their markets based on services (so-called *service value networks*). Business processes technology is used to prescribe how organizations work internally and how they work together to achieve the value of the service network. But the overall management of the corresponding business processes is growing more complex because of the inter-organizational and intra-organizational nature of business processes supporting the complex web of interactions of service value networks.

* 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).

© Springer-Verlag Berlin Heidelberg 2008

Several studies focus on creating and reconfiguring service value networks (see [1,3]). [1] proposes a methodology for analyzing the dynamics of value in networks at the operational, tactical, and strategic level with an emphasis on visualization and qualitative methods. In [2], the authors combine IT systems analysis with economic-based business modeling in order to build an e-business model that specifies e-business scenarios rather than on defining values. Besides the qualitative approaches, there is a growing need for quantitative methods. [3] presents a method for computing values by taking into consideration partners' satisfaction and additional value that is accrued by the relationship levels developed by the various partners.

In this paper we will focus our attention on *Service Networks* (SNs) (see [4,5]): it offers services that are obtained by composing other services provided inside the SN by a diversity of service providers by means of business processes.

From the operational view of the service network, one should focus on the management of the business processes and the monitoring of financial and operational measures of performance also called *Key Performance Indicators* (KPIs) in order to evaluate or improve them. Examples are overall process execution time, percentage of service requests fulfilling *Quality of Service* specifications, customer satisfaction index, etc. *Business Process Management* (BPM) together with Service Oriented Architecture (SOA) support organizations in the continuous improvement of their business's performance through the effective convergence of IT and business [6].

From the business view of the service network, there is a need to define the activities that achieve business goals such as cost cuts, market share increase, profit increase, customer satisfaction increase etc. Moreover, different partners may have different business goals, which may possibly be conflicting. For instance, one partner may be more interested in customer satisfaction, which may require an increase in costs to be achieved. This may be unacceptable for partners whose first priority is cost reduction. In [3] it is shown how the concept of value, properly defined, can be used as a unifying concept for studying service networks (called service value networks in that context) instead of the various heterogeneous business goals.

In this paper, we address the currently existing gap between business strategy and business models from one side and service system implementations on the other side. Strategic decisions (such as how to restructure the network; whether to leave a particular network to join another; or whether it is advantageous to join multiple networks at the same time; etc.) have to be made by the partners in order to increase their own value. Restructuring of a service network may be required to respond to competing networks or innovation in processes and technologies. Changes in the structure of the service network could drastically affect network partners' business objectives and/or network-wide business processes. Unfortunately, the current methods and tools for developing and managing service networks are highly fragmented, merely providing support for isolated parts of the huge task. This paper outlines a first attempt to develop and validate a comprehensive methodology for developing, monitoring, measuring and optimizing SOA-enabled business processes in SNs. We have developed the *Service Network Notation* (SNN) to represent participants in a SN and their interactions in terms of offerings and revenues. Such a comprehensive methodology is currently lacking. By adding SNs on top of the current BPM stack, analysts focusing on strategic goals of a business benefit from the detailed description and functionality of the business processes without being directly involved with BPM. This level of abstraction that is achieved through the linkage of SN to BPM provides them a better understanding of how to accomplish their goals.

The remainder of the paper is organized as follows. Section 2 introduces SNs through an example borrowed from the telecommunications industry. Section 3 introduces a meta-model of the SN. Section 4 shows how to analyze SNs and describe their basic properties. Section 5 describes standard BPM approaches, while section 6 proposes a novel architecture, SN4BPM, linking SN and BPM. Finally, section 7 provides some concluding remarks and discusses directions for future work.

## 2 SN by Example

In this section we describe the structure of service networks through an example taken from the telecommunications industry. Considering the methodology developed in [3], we model the service network of the telecommunications companies as a flow graph which comprises nodes (economic entities) and transfer objects (offerings which could be goods, services, information).

Our example is based on the Enhanced Telecom Operations Map (eTOM) [7] which is a reference framework for categorizing all the business activities that a service provider may use. In particular, we will describe the service network that is formed in order to set up a new service. We consider the following entities that collaborate with each other: the service provider (SP) offers services (realized as bundles of services such as orders for digital subscriber line, wireless, Internet data centre services, etc) to the subscribers. The external partners of the SP include the suppliers who provide resources (equipment, infrastructure, etc) and content providers with whom the SP co-operates in order to produce the bundle of services offered to the subscriber (e.g. video on demand, music educational content etc.) The internal partners of the SP (who can be outsourced and become external partners as well) are the call centre who provides information to subscribers over the telephone, the sales agent who provides prices for the different services to the subscribers, the service agent who is responsible for the set up and configuration of a subscriber's order, the field agent who performs service installations at the subscriber's site, the account manager who creates updates and manages accounts once the order is fulfilled and the billing agent who is responsible for the management of the billing system.

In Fig. 1, we provide a representation of the service network showing the relations created among the various entities. The economic entities are represented by circles and offering flows are represented through arcs. There are two types of offerings: services (depicted by solid arrows) and revenues (depicted by dashed arrows). A possible scenario for this example could be the following: A new subscriber contacts the call centre and orders the digital subscriber line service. The call centre enters the subscriber's information (name, address, etc.) to a customer information system and asks the sales agent to determine which services can be provided to this specific subscriber. The sales agent provides a list of possible services to the call centre which in turn informs the subscriber. The subscriber selects the service he wants and makes the order. The call centre submits the order to an order management system of the service provider. The account manager creates a new account for the subscriber and the service

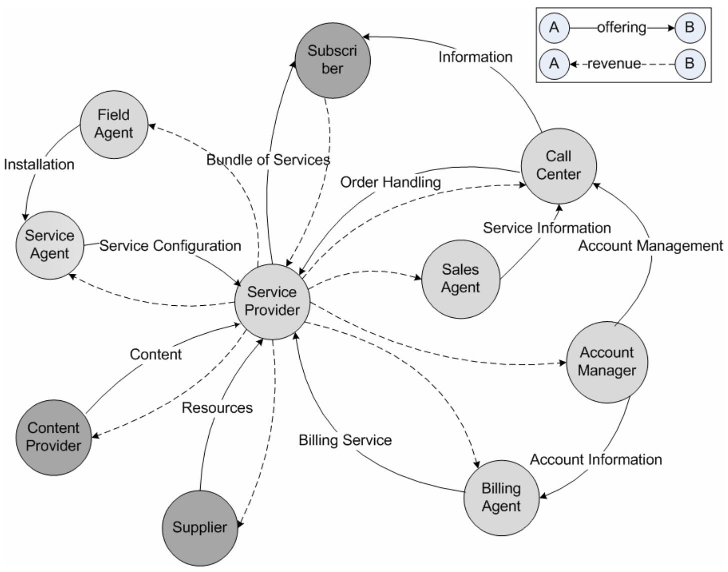


**Fig. 1.** The service network for a new service set up

agent configures the requested service and asks the field agent to install the equipment at the subscriber's site. As soon as the field agent completes his work, the service agent activates the new service.

The participants of the network, at the business level, are primarily interested in making sure that they derive value from their participation in the network. Participants in the network are also interested in promoting their own more general business objectives through their participation in the network, such as for example their market share, or their effectiveness in responding to market needs and being innovative, or their customer satisfaction. In section 4, we show how all these business objectives can be interconnected and also linked to IT level performance criteria such as SLAs, business processes, workflows performance etc.

## 3 SNN Meta-model

The meta-model for the SNN is shown in Fig. 2 as a UML2 Class Diagram. A Service Network consists of participants that are connected by relations. Participants and relations are represented by instances of the interfaces Participant and Relation. Instances of service networks, participants and relations have a name and are uniquely identified by an identifier. The interface Participant is implemented by the class Business Entity, representing providers and consumers of functionalities that generate value in a service network. SNN models comprise two kinds of relations: offering and revenue. Both kinds of relations connect a source and a target participant. Offering relations (modeled by the class Offering Relation) specify what services are offered (specified by the field offering) by the source participant (acting as service provider)

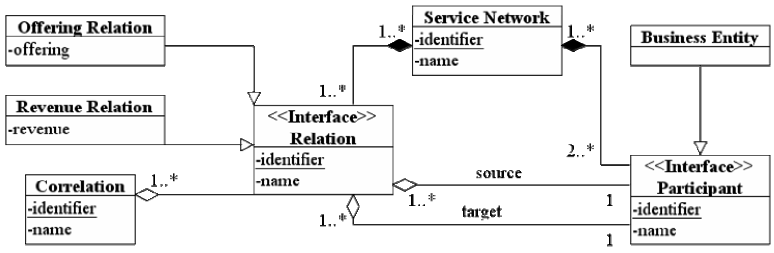


**Fig. 2.** A UML2 Class Diagram describing the SNN meta-model

to the target. Offerings could be goods or services, or a combination of both. Revenue relations (class Revenue Relation) describe the gain that the target participant has from the source in exchange for provided service. Revenues (modeled by the field revenue) are usually sums of money.

Generally, a SNN model describes interactions among a set of participants that take place over multiple, unrelated business processes. All the offering and revenue relations that take place over the same business process are *correlated*. Correlations allow to immediately visualize which parts of an SNN models pertain to a given business process, and which not.

## 4 Analysis of SNs

Organizations are expected to work worldwide fostering complex relations and developing complementary skills to generate and exchange goods, services or information. In order to evaluate and measure the performance of an organization within a service network and define business objectives as part of the firm's strategic behavior, the organization identifies specific KPIs [8]. Apart from measurements that take place at the BPM lifecycle (described in Section 6), KPIs are connected to parameters given in SLAs and parameters given by the interacting participants. For example, the value that a participant derives from the network is a KPI and could be connected, among other factors, to the satisfaction of this participant's customers. Satisfaction, in turn, depends on many factors such as the service delivery time, which usually should not exceed an upper bound specified in the relevant SLA.

To implement this service network, quite a few business processes must be deployed and operate such as: "order receipt", "order handling", "service configuration", "service installation", and "inquiries and complaint handling". These processes are distributed between several business units and business partners. To efficiently implement all these processes, SLAs will have to be agreed between partners. For example a cost KPI and cost reduction target for the SP will be affected by SLA requirements that a new service installation has to handled within a very limited timeframe, since the SP will have to pay service technicians and engineers to be available and on call to cover all new services requests by customers. Value derived from the network for, say, a content provider is affected by costs incurred for having sufficient equipment available to handle any real-time requests for content. On the other hand, if SLAs are not satisfied, then penalties for non-SLA compliance may have to be applied and customers' satisfaction may drop, thereby reducing the value derived for the content provider from its participation in the network.

It can therefore be seen that if business processes are implemented in sloppy and inefficient ways, or system and/or human resources are not used judiciously and are either wasted or under-provided, then the whole service network may break down, simply because the individual partners will not be achieving their desired KPIs and/or they will not be deriving sufficient value from their participation in the network. We now present elements of our modelling effort that tries to link satisfaction of business objectives and KPIs with SLAs and business process performance yardsticks.

The partners of a service network need to monitor on a periodic basis their KPIs and take corrective action as need be. The partners' job could be made significantly easier if they could use models that predict what the effect on a specific KPI, of a corrective action will be, and even better, what would be the optimal change (if this can be found) of parameter values and processes to yield the best possible change of a specific KPI. We are working on such models, and in what follows, we show how these models could be applied to our telecom' example to improve a specific KPI.

In our models, the KPIs are perceived as functions of all parameters that may affect their value. The shape of these functions can be affected by the structure of the business processes (for example, if the telecom provider in our example innovates and elimininates the need of technicians to install a new service, then a technician labor rate will obviously cease to have an effect on the function expressing the dependence of a cost KPI to various cost parameters). Let $\vec{x}_i = (x_{i1},\ldots,x_{iK}), i = 1,\ldots n$ be the input vector (e.g. services, resources, prices etc.) of a node (economic entity) $b_i$ that is used by the various functions expressing the KPIs of interest. For example, in the telecommunications example the vector $\vec{x}_i$ for the SP could be prices he imposes for the services he offers and the labor rates he pays to his employees. Consider now the function $f_i(\vec{x}_i)$ that denotes a KPI for $b_i$ due to its participation in the network. For example, this function could represent a revenues KPI, resulting from the sum of revenues of $b_i$, from all its network partners, to whom $b_i$ sells his services.

On the other hand, any prediction of improvement or even optimization of a KPI in our models, should also take into account constraints that exist. There are two forms of constraints: those that are intrinsic to the partner, such as maximum capacity of resources (number of people employed, maximum storage and CPU power available, etc.) and those that are imposed to the partner through the SLAs, for example maximum price tolerated by a partner's services buyer, or maximum delay tolerated for installing a new service in our telecom example, etc.

In general therefore, we can define the following maximization problem:

$$\max f_i(\vec{x}_i) \ s.t. \ \vec{x}_i < \vec{C} \tag{1}$$

where $\vec{C} = (C_1,\ldots,C_K)$ is the vector of constraints.

Next, we apply this framework to the telecommunications example. We choose to focus on value created for each partner, since this KPI has also been studied by us for other examples as well, see [3]. Though there are multiple ways to express value in models, we choose a relatively simple one: each participant captures value which is given by the sum of profits from interacting with nodes in a time interval and the expected value in the next time interval. The expected value of a participant represents the effect that all its relations have upon it and depends on the expected revenues of the next time period and on the expected degree of satisfaction that the participant's buyers have for his services.

How close is this representation of value to common practices in the marketplace? We claim that it is very close. The value of a business entity is usually estimated as the sum of several components, some of which are relevant to our service networks such as the profits of a business unit over a certain period (revenues minus costs) and the expectation of revenues over the next time period, and some of which are not related such as savings, capital equipment, etc. Notice also that estimating revenues is harder when a business unit is operating alone in the marketplace (its customer list being unpredictable and volatile) as opposed to when a business entity is operating within a network where buyers and sellers are fixed (at least for some period of time) and where customers tend to have long term relationships with their service providers. In such a network it is also feasible to get customers evaluations about the quality of their providers' services and integrate them into a "satisfaction index". Satisfaction index $Sat$ in our example is a function of the service delivery time, the price $p$ paid by the customer for the service, the requests/hour $n_1$ performed by agents, the number $n_2$ of customers that withdrew in the last period and the number $n_3$ of customers that complained in the last period. Although we give here simple examples of dependencies between the satisfaction index and the other parameters, empirical market studies can establish more accurate relationships.

Let us now apply the above ideas to our example and formulate a simple price optimization problem. We assume that calculations take place within a fixed time interval in which the network remains stable in number of participants. The value $V_{sp}$ of the service provider at the end of time interval $[T_{N-1}, T_N]$ as given in [3] is:

$$V_{sp}(T_N) = R_{sp}(T_N) - P_{sp}(T_N) + v_{sp}(T_N) \tag{2}$$

where $R_{sp}(T_N) = \sum_{i=1}^{n} p_i$ are the revenues by setting price $p_i$ for service type $i$, $P_{sp}(T_N) = \sum_{i=1}^{m} r_i$ are the payments by setting labor rate $r_i$ for type of employee $i$ and $v_i(T_N, Sat)$ is the expected value due to all the relations partner $b_i$ has in $[T_N, T_{N+1}]$. (For a more detailed description see [3].)

In order to calculate value according to equation 2 we need to calculate the above parameters. An upper bound on price $p$ and a labor rate $r$ are given in Service Level Agreements (SLAs) between the service provider and the customer and the service provider and his employees respectively. Response time $t$ is given in SLAs as upper bound and is calculated by the lower levels of the BPM layering stack. n and $n_1$ are calculated by the BPM layering stack and are used in order to calculate $t$. $n_2$ and $n_3$ are calculated by the BPM layering stack and are given together with $t$ and $n$ in the SN level in order to calculate the satisfaction and the value of the participants according to the equation 2.

In order to determine price $p$ such that the value of the service provider is maximized we solve the maximization problem given in equation 1 that is formed in the given example as follows:

$$\left.\begin{array}{l}\max V_{sp}(\vec{p}) \\ s.t.\ \vec{p} < \vec{p}_{SLA}\end{array}\right\} \Rightarrow \left.\begin{array}{l}\max(\sum_{i=1}^{n} p_i - \sum_{i=1}^{m} r_i + v_{sp}(T_N, Sat(t, p, n_1, n_2, n_3)) \\ s.t.\ \vec{p} < \vec{p}_{SLA}\end{array}\right\} \tag{3}$$

where $\vec{r}$ is a function of $\vec{p}$: $\vec{r} = g(\vec{p})$ and $\vec{p}_{SLA}$ is the upper bound of the price vector given in the SLA between the customer and the service provider. We assume that time $t$ is a parameter that is given to us by the analysis phase of the lifecycle described in section 6. We then calculate the price vector that maximizes value according to that price vector. In section 6 we will explain how this procedure enables the business analyst to adapt a changing environment to the participants' needs.

## 5 BPM Layering

From our study so far we have realized that in order to calculate KPIs and improve the performance of the network, we need to connect SN to BPM. For example, the response time depends on how business processes are performed and can only be calculated based on a detailed description of the corresponding business processes.

The currently accepted Business Process Management Layers will serve as a basis for the implementation/enactment of SNs. These different layers exhibit different levels of abstraction and different purpose of the models involved. The introduction of SNs as an additional layer on top of that stack has the goal of simplifying the procedure of modeling business processes that achieve strategic goals and hence reducing the gap between the business experts' view and the IT view on business processes. The extended BPM layering is shown in Fig. 3.

The process models layer contains process models defined in an abstract technology-independent manner. The target user group is mainly the group of business analysts. The processes are modeled in a coarse-grained manner - the main functional blocks are identified and connected, and no implementation details are specified here.

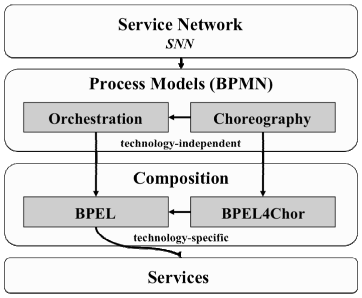


**Fig. 3.** Enhanced BPM Layering

This layer contains choreographies as well as orchestrations ([9], [10], [11]. The composition layer is the one with technology-specific definitions of process models. The target user group is the technical analysts. Both, choreographies and orchestrations are represented at this layer in terms of artifacts of a particular technology and refined and enriched with implementation-specific details [12], [13].

The service layer represents the set of available services that are exposed for use by the composition layer. The implementations of services are transparent, as well as the platforms on which they are deployed.

## 6 Enhanced BPM Lifecycle

In the BPM state of the art, the different techniques and technologies focusing on business processes are connected with each other by the *BPM lifecycle*, presented in Fig. 4 on the left. It comprises six phases: *analysis*, *modeling, IT refinement, deployment, execution* and *monitoring*.

The *analysis* phase consists of the elicitation of the requirements for the business processes. The *modeling* phase revolves around the design of abstract, high-level business processes (e.g., BPMN models, abstract BPEL processes) from the requirements gathered during the analysis phase. The abstract business process models, while not immediately executable, outline the overall structure of the final processes to a level of detail suitable to humans. Often during the modeling phase there are defects that emerge in the collected requirements. In such cases, the lifecycle reverts to the analysis phase in order to solve the issues. Abstract business processes models are transformed into executable process models during the *IT refinement* phase. The *deployment* phase deals with deploying on the enterprise information infrastructure the executable processes models produced in the IT refinement phase.

Once deployed, executable business process models enter the *execution* phase, where they are finally run. During their execution, processes instances produce events conveying information about executed activities, their performance, exceptions and faults that occur, and more. The events are collected and analyzed in the *monitoring* phase to adapt business process instances, measure KPIs, keep track of the overall state of the system, capture trends and patterns in the current usage of the processes, etc. The data processed in the monitoring phase are also taken into account in the analysis phase of the following iteration of the BPM lifecycle, providing feedback to evolve the business process models.

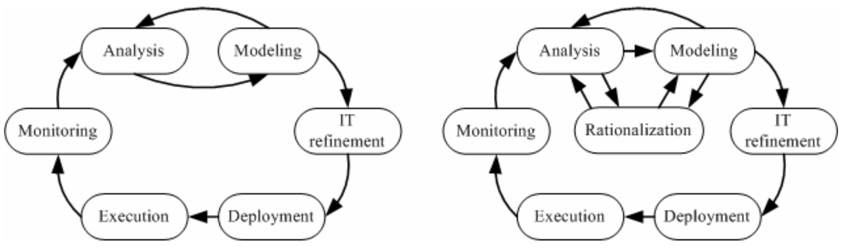


**Fig. 4.** The comparison between BPM lifecycle and enhanced BPM lifecycle

The canonical BPM lifecycle explained so far needs to be extended in order to benefit from the SNN and the analysis methods introduced in section 4. Fig. 4 (right side) presents the *Enhanced BPM Lifecycle*, obtained by adding a new phase, called *rationalization*, which deals with the modeling and analysis of SNN models.

The rationalization phase produces information which is used during either the modeling or analysis phase. We envision three ways of sequencing analysis, rationalization and modeling in the enhanced BPM lifecycle: *analysis–rationalization–analysis*, *modeling–rationalization–analysis* and *analysis–rationalization–modeling*. In the analysis–rationalization–analysis sequence (Fig. 5), the requirements resulting from the analysis phase are used in the rationalization one to create SNN models that represent the values flows among the participants. For example, the value calculation analysis described in section 4.1 is based on the requirements (e.g. an upper bound of the service delivery time) obtained from the analysis phase. The results are taken into account when modifying the abstract processes in order to maximize value. The new information on the desired characteristics of the process are then integrated with the previous set of requirements during another iteration of the analysis phase, during which takes place the resolution of conflicts that may arise between the original and new set of requirements.

In the modeling–rationalization–analysis sequence (Fig. 6), the existing abstract process resulting from the modeling phase is transformed into an SNN model through a *BottomUp transformation*. The value-maximizing analysis is then applied to the SNN model, producing a new set of requirements (e.g. a decreased upper bound of the service delivery time), which are integrated with the already existing ones in the upcoming iteration of the analysis phase. By analyzing SNN models extracted from abstract processes coming from outside the enterprise, it is possible to study the value flows from the point of view of the adopter of the processes and, for instance, take strategic decisions such as re-negotiate of the processes shared among participants.

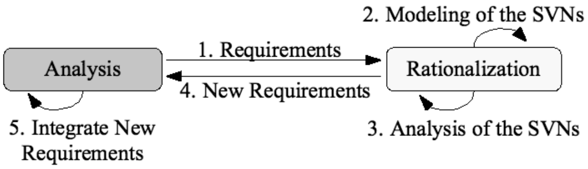


**Fig. 5.** The analysis-rationalization-analysis sequence

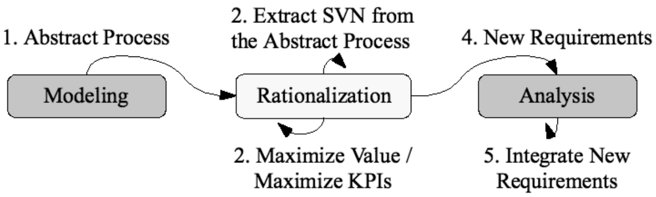

1. Abstract Process
2. Extract SVN from the Abstract Process
4. New Requirements
Modeling
Rationalization
Analysis
2. Maximize Value / Maximize KPIs
5. Integrate New Requirements

**Fig. 6.** The modeling-rationalization-analysis sequence

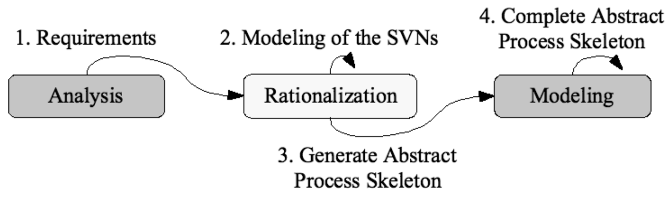

1. Requirements
2. Modeling of the SVNs
4. Complete Abstract Process Skeleton
Analysis
Rationalization
Modeling
3. Generate Abstract Process Skeleton

**Fig. 7.** The analysis-rationalization-modeling sequence

In the analysis–rationalization–modeling sequence (Fig. 7), the requirements resulting from the analysis phase are used in the rationalization phase to realize one or more SNN models. These models are transformed into *abstract process* models by applying *TopDown* transformations. The transformations use the correlations among offering and revenue relations to define the boundaries of the conversations involving the participants in the service networks.

The analysis–rationalization–modeling and modeling–rationalization–analysis sequences create a bond between SNN models and the abstract process models developed during the enhanced BPM lifecycle. Revenue and offering relations connecting parties in SNN models are translated into conversations and interactions in the abstract processes. Changes to SNN models (i.e., the removal of a revenue relation) can be mapped, through changes in the requirements, to changes to be applied to the abstract processes.

## 7 Conclusions and Future Work

Currently, we are witnessing an evolution in service oriented economies that need technological means to support them. In this paper we propose an architecture to coordinate business processes lifecycle and bridge existing gaps between technical and business perspectives. Our approach provides an abstract way to support business processes (in the SN level) and conversely a detailed description of the service network (in the BPM level). Next, we aim to formulate variations of optimization problems involving different kinds of KPIs and SLAs. The behavior of competing networks is also an open problem to be addressed possibly through means of game theoretic concepts. In this context, as interaction among different business roles in the process of providing a service is a key element in understanding and observing service systems, the field of game theory becomes a useful tool for identifying rules and strategies that optimize business objectives. As it was already done, all these studies have to be linked to the lifecycle management of business processes so that any progress made at the optimization level can be exploited by the business analysts.

## References

1. Verna, A.: Reconfiguring the Value Network. Journal of Business Strategy 21(4) (July-August 2000)
2. Gordijn, J., Akkermans, H.: Designing and evaluating e-business models. IEEE Intelligent Systems 16(4), 11–17 (2001)

3. Caswell, N., Feldman, S., Nikolaou, C., Sairamesh, J., Bitsaki, M., Koutras, G.D., Iacovidis, G.: Estimating Value in Service Systems – A theory and an example. IBM Systems Journal 47(1) (2008)
4. Sampson, S.E.: Understanding Service Businesses: Applying Principles of Unified Services Theory. Wiley Press, Chichester (2001)
5. Spohrer, J., Maglio, P., Bailey, J., Gruhl, D.: Steps Towards a Science of Service Systems. Computer 40(1), 71–77 (2007)
6. 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)
7. Enhanced Telecom Operations Map The Business Process Framework For The Information and Communications Services Industry, TeleManagement Forum (2003), `http://www.tmforum.org`
8. Neely, A., Gregory, M., Platts, K.: Performance measurement system design: A literature review and research agenda. International Journal of Operations & Production Management 25(12), 1228–1263 (2005)
9. Business Process Modeling Notation (BPMN) Specification, Final Adopted Specification. Technical report, OMG (Feburary 2006), `http://www.bpmn.org/`
10. Keller, G., Nüttgens, N., Scheer, A.-W.: Semantische Prozessmodellierung auf der Grundlage Ereignisgesteuerter Prozessketten (EPK). Technical Report Heft 89, Universität des Saarlandes, Veröffentlichungen des Instituts für Wirtschaftsinformatik, IWi (1992)
11. Zaha, J.M., Barros, A., Dumas, M., ter Hofstede, A.: A Language for Service Behavior Modeling. In: CoopIS, Montpellier, France (November 2006)
12. Leymann, F.: Web Services Flow Language WSFL. IBM Corporation (2001), `http://www.ibm.com/software/solutions/webservices/resources.html`
13. Thatte, S.: XLANG: Web Services for Business Process Design. Microsoft Corporation (2001), `http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.html`

# Ad-Hoc Usage of Web Services with Dynvoker

Josef Spillner, Marius Feldmann, Iris Braun, Thomas Springer, and Alexander Schill

Dresden University of Technology, Dept. of Computer Science, Chair for Computer Networks, 01062 Dresden, Germany
{josef.spillner,marius.feldmann, iris.braun,thomas.springer,alexander.schill}@tu-dresden.de

**Abstract.** While web services are often targeted at machine-to-machine communication, they are also increasingly used directly in the interactions between humans and machines. Instead of developing specialised client applications for the invocation of these services, a generic human-driven ad-hoc usage is beneficial in many scenarios, including rapid service testing and dynamic inclusion of services as plugins into applications. We argue for the need for such a usage and extract requirements for generic web service clients. We then present a few selected use cases and introduce the Dynvoker client which already passes the majority of evaluation criteria. With its technical capabilities and open and vivid development, we consider it the most suitable and flexible generic client available and therefore highlight its role as a central component in a user-centric web service research project.

## 1 Introduction

Rich and thin client applications provide a human interface to computational functionality. In rich client applications, the interface and the functional part are tied together, contrasting the rather loose coupling in thin clients. Some functional parts are designed to allow primarily programmatic access and provide an API over the network. Among these are Web Services, which are in most cases self-described, stateless components. Sometimes, only an informal, textual description of the interface exists, and the provider offers custom-made toolkits to foster client development. Nevertheless, there are widely used file format specifications to describe aspects like the message syntax, operational semantics and non-functional properties. When these descriptions are present, it is possible to call the services with generic clients by introspecting the descriptions and deducing behavioural information.

There are several use cases where combining evolving services with existing applications can benefit from ad-hoc usage. In applications with plugin support, many plugins rely on a specific service interface. With automatically generated forms to access the service, the service can evolve and be improved without the need to transfer a new GUI component to the client. Furthermore, once a form generator for a GUI technology has been developed, it allows the access to

© Springer-Verlag Berlin Heidelberg 2008

all existing services without the need for cooperation from the service authors. Even if a custom client is going to be developed, a generic client can assist in rapid functional tests. Generic clients are also useful in mobile scenarios to avoid development and installation of custom clients [1].

The process of generating the user interface based on various information from the service provider and enabling an interaction between the user and the service consists of a number of steps which are being researched in the area of WSGUI, or *Web Services Graphical User Interfaces* [2]. Dynvoker, a dynamic service explorer and invoker, is an implementation of these concepts, generally called generic web service client or more technically WSGUI engine. Compared to existing approaches, we contribute such a WSGUI engine which accepts multiple web service description formats as its input and can generate user interfaces and forms in various output formats. This design choice leads to greater user experience by offering a higher number of services on a higher number of devices, but also presents some challenges in handling the differences between the formats. We will show that this design choice is superior to single-format implementations and will outline results from practical experience with our open-source implementation.

The structure of this paper is as follows: First, related work is evaluated, concentrating on run-time user-centric web service interaction tools. Afterwards, a requirements analysis of features common to most of these approaches is performed, followed by a number of features in Dynvoker which help to fulfil the requirements. The text concludes with a brief report on process handling and an outlook on how Dynvoker will be used in an existing research project.

## 2 Related Work

Generating user interfaces dynamically to access a well-defined interface or an underlying data model has been in the focus of research for several years. Central ideas used in UI generation for web services have been extracted from similar fields of research and apply as well to areas such as automatic dialogue generation from the underlying configuration schema[1] or inferring user interfaces from database models [3]. However, the specific field of ad-hoc usage of web services by automatically created UIs is only unsatisfactorily covered by research and development projects. Though a lot of preliminary work has been invested into this area, only a few implementations are still available. An early approach that clearly formulated the intention to dynamically integrate web services into a user interface has been the web service User Interface (WSUI) initiative that had the goal to embed services as visual components into web portals. The initiative has stopped its activities shortly after its foundation. Neither the website nor the specification draft are still available.

A further historic approach which forms an important building block for current projects is the original WSGUI project [4] which influenced Dynvoker in many aspects. Besides inferring basic information about the user interfaces from

[1] KXForms dialogue generation: `http://www.lst.de/~cs/kode/kxforms.html`

the web service it introduced the annotation format GUI Deployment Descriptor (GUIDD)[2] that enables aspects such as attaching multilanguage human-readable labels to input or output fields. After merging inferred information and the optional GUIDD data, the resulting form based screen data was transferred via XSLT to a concrete user GUI representation.

The open source library Xydra[3] can be used for ad-hoc creation of UIs for web services. It produces XHTML files with web forms based on an inference mechanism for WSDL and associated XML Schema. Besides describing service annotations based on ontologies it employs a technique called TreePath to be able to represent arbitrary XML structures as key-value pairs required by XHTML browsers. The project development was stopped in 2003.

Further purely inference-based mechanisms are the Dynamic SOAP Portlet[4] and the SOAPClient[5]. Whereas the first one follows a portlet concept that dynamically offers a UI for a generic client for web services [5], the second one can be seen as a testing tool for web service development. It creates on-demand a rudimentary HTML form for all operations found within a WSDL file specified by the user. There is no information available about the interior of this application.

Some common development tools offer support for web service UI creation. An advanced implementation is the XML Forms Generator[6] available as Eclipse plug-in. Though it does not fit into the category of ad-hoc UI generation, it offers interesting concepts relevant for the on-demand creation process as well. It analyses a WSDL document and enables combining derived data with an Eclipse Modelling Framework model like an XML Schema file or a UML diagram for providing information such as type information. The tool generates an XHTML output with associated CSS style sheet. For REST-based interfaces described by WADL, the NetBeans IDE[7] offers a forms inference tool for testing services during the development time.

Academic publications covering the topic of ad-hoc UI generation for web services are rare. [6] directs a focus on dynamic creation of multimodal UIs using XForms and VoiceXML elements generated from WSDL inference. The transformation to concrete UI representations is based on XSLT. Though it is pointed out that service descriptions can be imported to a system specific proxy server for providing additional information to improve the quality of the UI, no details about this possibility are given. Furthermore, various future research intentions are mentioned though none of them have been realised yet. In [7], a further system for UI generation at runtime is proposed using four different WSDL annotation files containing UI related information. The system supports a profile-driven adaptation to different user-clients. No arguments are provided

[2] GUIDD specification: `http://wsgui.berlios.de/guidd/`
[3] Xydra generic client: `http://www.extreme.indiana.edu/xgws/xydra/`
[4] Dynamic SOAP Portlet: `http://soap-portlet.sourceforge.net/`
[5] SOAPClient: `http://soapclient.com/`
[6] XML Forms Generator: `http://www.alphaworks.ibm.com/tech/xfg`
[7] NetBeans: `http://www.netbeans.org/`

for the chosen system architecture. The different WSDL annotation formats are not described in any detail beyond their overall focus.

The mentioned approaches do not offer a generic solution covering various service interface descriptions such as WADL or WSDL at once. All of them are bound to concrete service technologies. Furthermore, XSLT is a quite common means to realise the transformation to concrete UI representations, although the difficulties regarding its complexity are well-known. Only a few of the analysed projects are still active and offer a directly testable implementation. Despite some of them providing basic information about the overall mechanisms, they mainly do not provide any internal details.

## 3 Aspects of Ad-Hoc Usage

Ad-hoc usage of simple services requires at least navigation to find the desired service, form generation and submission as functionality. While submission is done in the background and involves the interaction with a service, navigation and form generation involve the user and are therefore of interest to us.

### 3.1 Navigation to the Service

Navigation guides the user from the expression of a goal to the input form, which is generated automatically. After the submission of the form, the service is invoked and the output form is rendered based on the results. All of these steps bound together form an interaction pattern. For simple cases, the goal would be expressed as a direct link to the service description file as shown in Fig. 1. For more advanced cases like interacting with processes consisting of many services, like selecting a service from a registry first before using it, a more sophisticated interaction pattern needs to be defined. Interestingly, it could be derived from a formal process model, too. The relationship between navigation, form generation, submission and interaction in such an advanced case is represented in Fig. 2. It is worth mentioning that forms can either be pure input and output forms, or be of a hybrid nature, using previous input or output information to pre-fill parts of the input form.

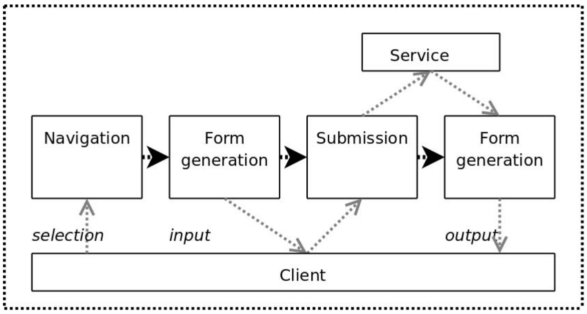


**Fig. 1.** Basic interaction model for ad-hoc service usage

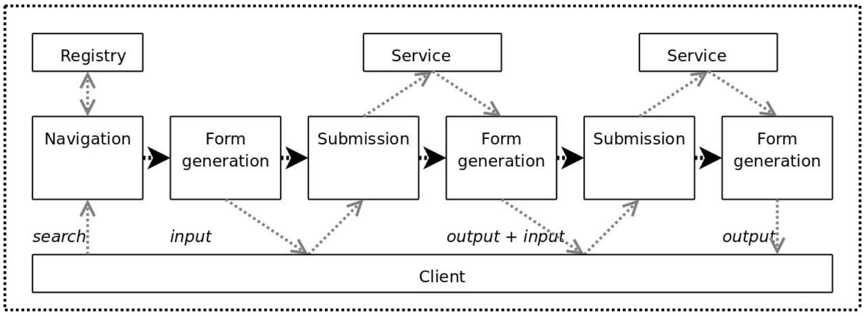


**Fig. 2.** Complex interaction including search and orchestration

## 3.2 Form Generation

Form generation is a traditional topic in the model-driven and human factors communities. Like most other approaches, we are focusing on the generation itself and do not currently evaluate usability concerns, although we acknowledge their importance for acceptance with users.

A number of individual steps have to be performed in order to achieve a suitable form. Among them are the creation of form elements, layout and embedding the form into an application context like a desktop dialogue or a website.

An additional requirement for practical use is that it should be possible to augment existing services with local hints for the graphical representation. This does not exclude an approach which integrates such hints with the service description, but allows for a greater independence from service providers.

In summary, we have identified the following technical requirements for a generic web service client:

– Ability to understand a variety of web service description formats, with or without integrated hints for graphical representation
– Ability to load external graphical, textual and semantic hints
– Ability to generate user interfaces in a variety of formats, either abstract or concrete
– Ability to define interaction models to not limit the engine to a single web service invocation
– Complete and correct visual representation of the programmatic interface

## 4 Dynvoker Approach

Following the discussion of requirements, this section is presenting the features and implementation of Dynvoker as a generic web service client. Before delving into the feature set, the overall architecture is briefly presented in Fig. 3. Dynvoker consists of a relatively small application core which can be run as a servlet, a web service or a command-line application. The generic handling of

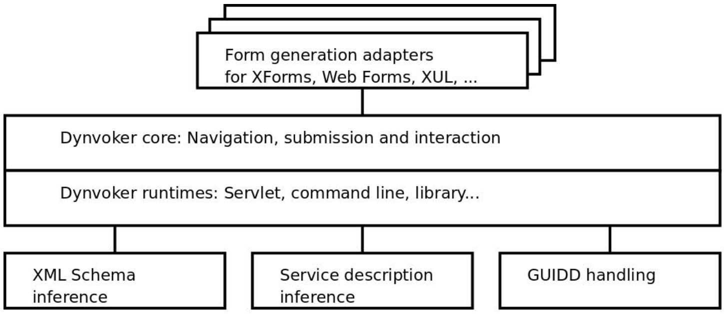

Form generation adapters
for XForms, Web Forms, XUL, ...
Dynvoker core: Navigation, submission and interaction
Dynvoker runtimes: Servlet, command line, library...
XML Schema inference
Service description inference
GUIDD handling

**Fig. 3.** Overall architecture of Dynvoker

input, i.e. web service descriptions, and output, i.e. user interfaces, is reflected in the modular architecture. It contains several adapters to generate forms, and inference modules for various service description formats.

## 4.1 Inference from Web Service Descriptions

In order to use web services without prior knowledge of their expected input or behaviour, it is necessary to infer this knowledge from the service description. Knowledge about the service methods, parameter names and structure can usually be derived from it. We have previously reported on details and issues of inference of user interfaces from XML Schema [8] and will therefore concentrate on the nature of inference from generic service description formats. The dominant description format is WSDL 1.1, which is used mostly for method-centric, i.e. SOAP-based services, although its successor WSDL 2.0 also contains bindings for resource-centric, i.e. REST-based services. However, alternative formats like WADL, the Web Application Description Language, exist for generic REST-based services, and even specialised formats like OpenSearch [9] for the specific domain of REST-based search services. Both WSDL and WADL use XML Schema to define the structure of the messages or resource representations, whereas OpenSearch is limited to formatted query URLs for the input and extended RDF for the output.

Dynvoker is able to infer the contents of a service, like the operations or resources it offers, from WSDL and WADL files, and will generate output which lets the user navigate to the service of choice and select the appropriate service. When a WADL-described service is chosen, the service selection interaction is extended by offering a number of resources for each method. Input and output forms are generated based on the XML Schema. The generation architecture is shown in Fig. 4.

Alternative service descriptions can be supported through transformations. OpenSearch descriptions are converted to WSDL first and can then be handled as usual without additional code. D-Bus, the dominant application-level interprocess communication (IPC) system on Linux, provides its own IDL-like method

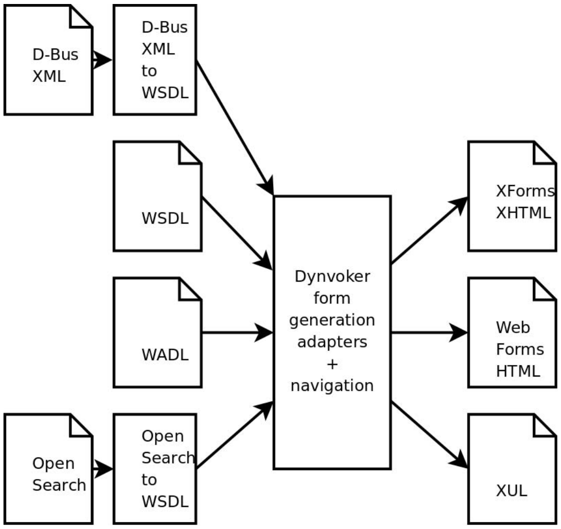


**Fig. 4.** Inference transformation process

description format which can be retrieved through service introspection. We have implemented a bidirectional gateway between web services and D-Bus, which works independently from Dynvoker, to prove our claim.[8] Since WSDL provides a superset of the service description abilities of D-Bus, the conversion always works in the direction we need for Dynvoker.

## 4.2 Additional GUI Hints

Automatically generated user interfaces are at risk of providing inferior quality and usability than manually designed ones, depending on the completeness of the information in the model or any web service description. On the other hand, a strictly rule-based design leads to consistent interfaces which can completely encompass the service functionality and automatically adapt to evolving services, including the alteration of message formats [10][11].

Therefore, as many aspects of the generation process as possible should be configurable without endangering the consistency and completeness properties. The amount of hints needed decreases with the expressiveness of the service description format. For common WSDL-described services, Dynvoker can use GUIDD files containing semantic hints, UI hints and UI services to improve the resulting forms, as shown in Fig. 5.

*Semantic hints* are useful in combination with purely syntactical description formats like WSDL to yield more appropriate user interfaces. For example, the only inferable information about the password field in Fig. 7 is that it is of type `string`. To avoid a free-form input field and use a special password entry field

[8] D-Bus Web Service Proxy: `http://techbase.kde.org/Projects/D-Bus-WS`

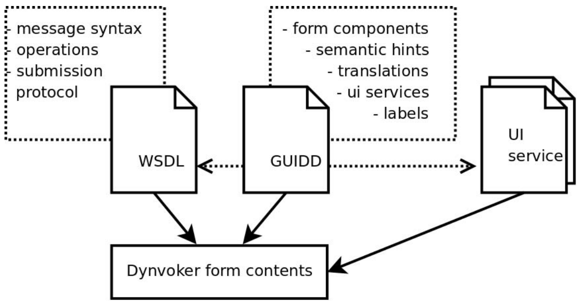


**Fig. 5.** Information sources containing additional GUI hints

instead, a semantic hint is added and will result in password fields independent of the output format.

*UI hints* include labels with translations, frame captions and substitutes for otherwise auto-generated fields, so-called form components. As opposed to semantic hints, they depend on the resulting output format. For web-based interfaces, style sheets can be used to give form components a consistent look and feel. UI hints for abstract user interfaces are also possible and are discussed in the evaluation part.

*UI services* represent a novel concept which lifts explicit GUI hints to a service-oriented level. This lifting makes it possible to exchange the hints or the providers

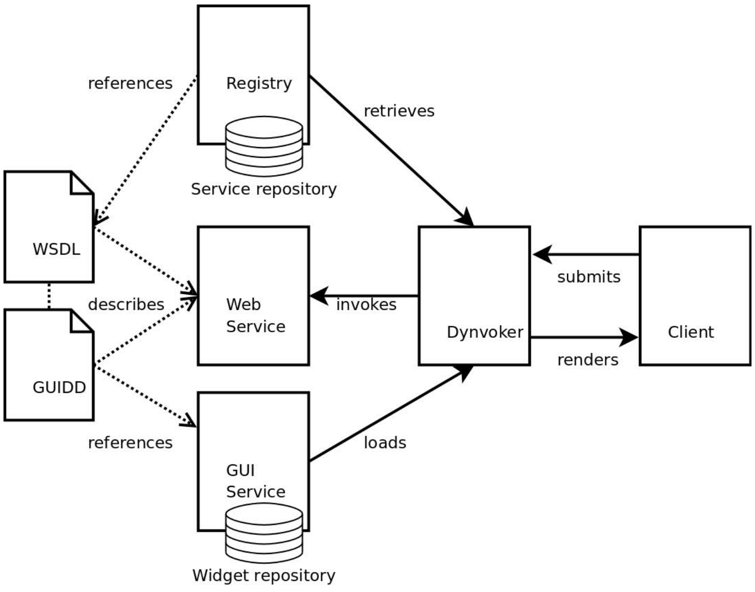


**Fig. 6.** UI services and web services in dual use

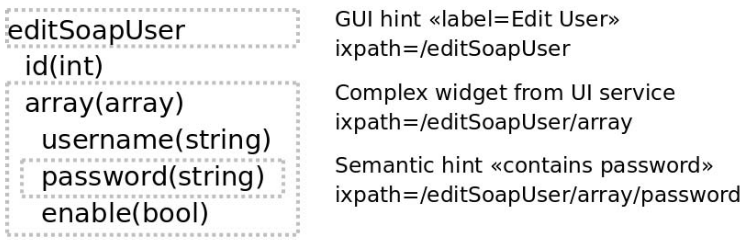

editSoapUser
id(int)
array(array)
username(string)
password(string)
enable(bool)
GUI hint «label=Edit User»
ixpath=/editSoapUser
Complex widget from UI service
ixpath=/editSoapUser/array
Semantic hint «contains password»
ixpath=/editSoapUser/array/password

**Fig. 7.** Hint locators in a GUIDD file applied to a SOAP message instance

of the hints, therefore driving the customisation of applications. Our implementation of UI services is based on a widget repository with query interface for Dynvoker and a submission interface for UI widget designers as shown in Fig. 6. A widget connector within Dynvoker searches for available widgets and renders them into form components, aligning the further processing with UI hints. This includes a distinction between simple and complex UI hints, the latter ones covering complex types like lists.

All of these three groups of hints are stored in the already mentioned GUIDD files. If they are passed to Dynvoker, the generated forms can be improved. The reusability of GUIDD files, especially in combination with reusable data schemas in WSDL files, helps in further advancing the acceptance of SOA by eliminating redundant client development.

In Fig. 7, the locator mechanism for interleaving GUI hints for a user management operation in the SOAP API of the Asterisk telephony server is shown. GUIDD uses higher-level schema and instance XPath expressions which are aligned with the reusability of schema components. In the figure, the semantic hint for the password entry field collides with the UI service for the complex user data type `array`. The use of a GUIDD editor can help avoiding such collisions.

## 4.3 Process Integration

Up until now we have assumed the interaction between a user and a single service in our explications. This is not always sufficient in dynamic service landscapes with complex interactions between humans and processes.

We have previously proposed the WSInterConnect distributed architecture to integrate humans into processes based on interactions with Dynvoker [12]. The industry proposal BPEL4People/WS-HumanTask was already mentioned as a potential hook for this distributed architecture and has matured since then, but implementations are still not widely and freely available. Major flaws of this extension include an insufficiently specified visual representation of messages to the user and a lack of process launching interaction. A Dynvoker-based approach named Unified Process and Task Management Interface, or UPATMI, is currently being developed by us to solve this problem.

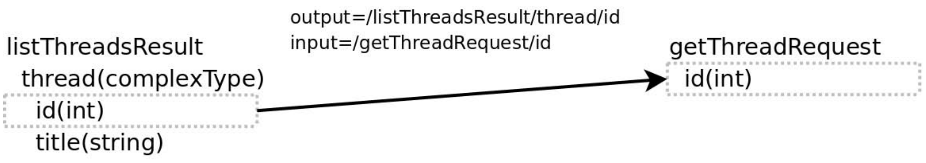


**Fig. 8.** Links between service operations

Another approach is to reduce the number of interactions needed with a process by inspecting it, essentially treating it as a grey box, whenever possible. The Dynvoker variant GUI4CWS has proven it to work for a subset of BPEL [13].

Finally, light-weight links between service calls without the need for an executed process were implemented as a GUIDD extension. This makes it possible to implement interactive applications with purely declarative syntax. For example, the list of topics in a forum as output message of the default operation `listThreads` would add a link to each thread which invokes Dynvoker with the operation `getThread` like shown in Fig. 8.

## 4.4 Status of the Resulting Implementation

Dynvoker has been developed for about two years now, entailing a number of improvements in a still ongoing process. On the other hand, it has uncovered a number of weaknesses in existing standards and implementations especially for XML Schema and XForms. This section compares the current implementation with the list of requirements, reasons about deviations and confirms the necessity of some of the assumptions we made.

Abstract user interface languages are currently not supported, but the Dynvoker architecture allows the creation of new output adapters for such languages. The resulting forms could then be displayed in applications which can render them, or convert them for display on legacy applications. This approach can also be followed with the existing XForms adapter by converting the output to HTML with JavaScript. However, according to our tests, even advanced tools like Chiba do perform this task correctly, as can be verified by anybody by selecting this transformation mode on the Dynvoker website. Therefore, we focused on writing adapters for concrete UI languages, but appreciate the potential of abstract languages.

We have not yet implemented the interaction models as executable processes within Dynvoker. All interaction patterns are currently hard-coded. We strive to add this in a future version based on GUI4CWS.

All the remaining requirements we have outlined are already supported by Dynvoker. In particular, the ability to use both resource-centric and method-centric web services contributes to hiding protocol details from the user. Additional GUI hints are supported in a way that the correctness and completeness properties from the inference mechanism will not be violated.

A large number of services with WSDL and WADL descriptions can already be used with Dynvoker. For those services for which a GUIDD exists, the user experience is clearly better than for those without. We follow a live validation approach where any interested person can verify our results on the Dynvoker portal.[9]

## 5 Summary and Future Steps

Building up on previous detailed analysis of issues in ad-hoc service usage, we have shown that Dynvoker is a viable generic client which solves many of the issues. None of the alternative approaches can dynamically explore method-centric and resource-centric services alike, output forms in various formats or integrate GUI services to provide a richer user experience. The generic design of many parts of Dynvoker has yielded a lightweight architecture which is freely available to any interested person as an open source project.[10]

In the future, we expect to integrate even more process-related functionality and add collaboration methods to the Dynvoker portal to help building communities of users of explorable services. Furthermore, a major focus will be directed to the optimisation of UI design for complex web services, especially in the dimension of usability by solving partial aspects during a design-time stage. Its central goal is to create a model-driven service engineering methodology supported by design-time concepts and tools for the development of client applications for single and composed web services. Due to the obvious fact that some aspects such as dynamic binding of concrete services and runtime optimisation are not feasible during design-time, we aim to define a runtime platform for handling these and further runtime dynamic concerns within the ServFace project.[11]

## References

1. Sánchez-Nielsen, E., Martín-Ruiz, S., Rodríguez-Pedrianes, J.: Mobile and dynamic web services. In: Proceedings of the ECOWS 2006 Workshop on Emerging Web Services Technology, Zurich, Switzerland (December 2006)
2. Spillner, J., Braun, I., Schill, A.: Flexible human service interfaces. In: Proceedings of ICEIS. Volume HCI. International Conference on Enterprise Information Systems (ICEIS), Funchal, Madeira - Portugal, pp. 79–85 (June 2007)
3. Bajaj, A.: Inferring the User Interface from an EER Data Schema. In: Proceedings of the Americas Conference on Information Systems (AMCIS), paper 471, Acapulco, Mexico (August 2006)
4. Kassoff, M., Kato, D., Mohsin, W.: Creating GUIs for Web Services. IEEE Internet Computing 7(4), 66–73 (2003)
5. Gesser, C.E.: Uma abordagem para a integraçã dynâmica de servios web em portais. Master's thesis, Universidade Federal de Santa Catarina (2006)
6. Steele, R., Khankan, K., Dillon, T.: Mobile web service discovery and invocation through auto-generation of abstract multimodal interface. itcc 2, 35–41 (2005)

[9] Dynvoker portal: `http://dynvoker.org/`
[10] Dynvocation research project: `http://dynvocation.selfip.net/`
[11] ServFace website: `http://www.servface.eu/`

7. He, J., Yen, I.L.: Adaptive user interface generation for web services. In: Proceedings of the IEEE International Conference on e-Business Engineering, Hong Kong, China (October 2007)
8. Spillner, J., Schill, A.: Analysis on inference mechanisms for schema-driven forms generation. In: Tagungsband XML-Tage, Berlin, Germany, June 2007, pp. 113–124 (2007)
9. LeVan, R.: OpenSearch and SRU: Continuum of searching. Information Technologies and Libraries (ITAL) 25(3), 151–153 (2006)
10. Trapp, M., Schmettow, M.: Consistency in use through model based user interface development. In: The Many Faces of Consistency in Cross-Platform Design at CHI 2006, Montréal, Québec, Canada (April 2006)
11. Nichols, J., Chau, D.H., Myers, B.A.: Demonstrating the viability of automatically generated user interfaces. In: Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 1283–1292 (2007)
12. Spillner, J., Braun, I., Schill, A.: WSInterconnect: Dynamic composition of web services through web services. In: Eliassen, F., Montresor, A. (eds.) DAIS 2006. LNCS, vol. 4025. Springer, Heidelberg (2006)
13. Bleyh, N.: Analyse und Vergleich von Ansätzen zur Einbindung von menschlichen Interaktionen in komplexe Web Services. Master's thesis, TU Dresden (June 2006)

# A Web Services Gateway for the H2O Lightweight Grid Computing Framework

Mauro Migliardi

CIPI – Universita' di Genova
Via Opera Pia 13
16145 – Genoa
Italy

**Abstract.** H2O is a lightweight distributed component framework for the dynamic aggregation of software components, services and computational resources into Grid Computing Systems. H2O provides a powerful separation of roles clearly distinguishing providers of software components from provides of computational services, this model allows developers to easily design layered applications and to deploy them on top of dynamically aggregated computational nodes. The ease of use does not exact a weak security system, in fact, by combining the native Java sandbox model and the use of JAAS, H2O provides a robust security layer. Although H2O supports an extended version of Java RMI (RMIX) as its native inter-component communication language, the software components deployed inside an H2O virtual machine are exposed only as Java Objects. In this paper we present the H2O Web Services Gateway, a set of H2O software component capable of dynamically capturing the deployment of new software components into an H2O virtual machine and automatically generating and publishing the WSDL description of these components. This feature, combined with the use of the Web Services Invocation Framework, enables the automated export of software components deployed into an H2O virtual machine as Web Services and facilitates the integration of lightweight Grid application into Service Oriented Architectures.

**Keywords:** Grid-Computing, Web Services, Service Oriented Architecture, Software Components Frameworks.

## 1 Introduction

H2O is a lightweight distributed component framework for the dynamic aggregation of software components, services and computational resources into Grid Computing Systems [1][2][3][4]. H2O provides a powerful separation of roles clearly distinguishing providers of software components from providers of computational services; this model allows developers to easily design layered applications and to deploy them on top of dynamically aggregated computational nodes. The ease of use does not imply a weak security system, in fact, by combining the native Java sandbox model and the use of JAAS [5], H2O provides a robust security layer. H2O supports an extended

© Springer-Verlag Berlin Heidelberg 2008

version of Java RMI (RMIX [4][6]) as its native inter-component communication language. This technology implements a powerful pluggable mechanism that allows transporting an almost complete set of Java RMI call semantics on top of an extensible set of transport protocols. The RMIX included in the standard H2O distribution supports transport protocols such as Java RMI and SOAP. However, while this pluggability allows using RMI style programming on top of heterogeneous protocols, H2O currently provides mostly client side modules, thus the software components deployed inside an H2O virtual machine are still exposed only as local or remote Java Objects. Furthermore, while H2O is designed to be easily extended with resource monitoring facilities both local to each node and spread over distributed virtual machines [1], the standard H2O distribution does not include any form of resource monitoring service, thus it is not possible to automatically discover and manage new services.

In this paper we present the H2O Web Services Gateway, a set of H2O software component capable of dynamically capturing the deployment of new software components into an H2O virtual machine and automatically generating and publishing the WSDL description of these components. This feature allows using standard Web Services client side invocation semantics and software, such as the Web Services Invocation Framework [7], to access software components deployed into an H2O virtual machine. Thus the H2O Web Service Gateway enables the automated export of software components deployed into an H2O virtual machine as Web Services and facilitates the integration of lightweight Grid application into Service Oriented Architectures.

This paper is structured as follows, in section 2 we provide a brief description of the H2O framework, its main characteristiques and the capabilities of its native inter-component communication protocol RMIX; in section 3 we describe the software components that combine into the Web Services Gateway; finally, in section 4 we provide some concluding remarks.

## 2 The Framework

In this section we provide a brief introduction to the H2O framework and RMIX, its native inter-component communication protocol. The description provided in this paper is extremely introductory any further detail can be found in [1], [2], [3], [4] and [6].

### 2.1 H2O

H2O is a Meta-computing project designed and developed at the Distributed Computing Laboratory of the Emory University of Atlanta.

It takes origin from the HARNESS [8] project developed in the end of 1990s in the same laboratory. H2O is a Java-based middleware implementing the concept of services container in a totally distributed way. H2O is designed to provide support to the plug-in based distributed virtual machine model first described by the HARNESS system [9]. H2O is explicitly designed to be a secure, scalable, stateless, and lightweight middleware for distributed applications. Conceptually, H2O is a distributed

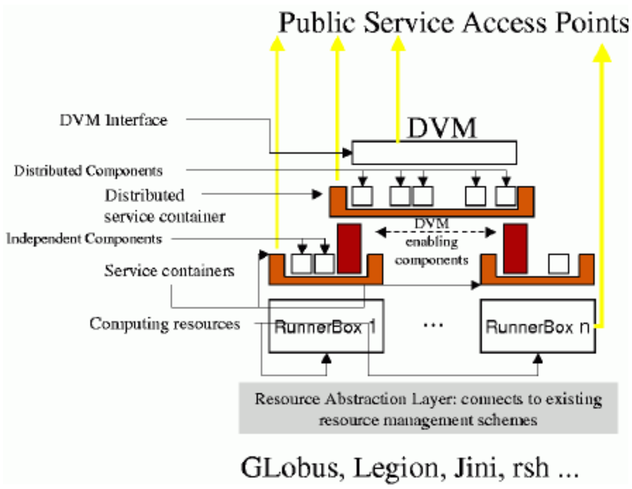

Public Service Access Points
DVM Interface
DVM
Distributed Components
Distributed service container
Independent Components
DVM enabling components
Service containers
Computing resources
RunnerBox 1
...
RunnerBox n
Resource Abstraction Layer: connects to existing resource management schemes
GLobus, Legion, Jini, rsh ...

**Fig. 1.** Layered Architecture of the H2O Framework

component framework [10], however it distances itself from mainstream technologies and frameworks such as J2EE [11] as it removes the static binding between service deployment/deployer and resource provision/provider. The services deployment can be done by any third part or client, and it is clearly not required that it is performed by the resource owner. Hence, resource sharing and grid computing systems can be built up using H2O in a very lightweight fashion.

H2O provides Java APIs for remote component deployment and management, and inter-component communication. H2O components can communicate via remote method invocations (both synchronous and asynchronous), and through a publisher-subscriber distributed event model. The protocol-related communication aspects are cleanly separated from the application code, making development of secure, distributed applications simple and efficient. With H2O it is possible to build Grid and Parallel application using Java language avoiding completely the adoption of classical parallel programming paradigm as MPI or PVM. Even if the framework does not provide any explicit support with parallel libraries, the set of communication protocols together with deployment and management capabilities provided by the platform make easy to deploy Grid application. Moreover, the administrative management of the whole distributed infrastructure is totally equivalent to a common Java application. H2O may support wide range of distributed programming paradigms, including self-organizing applications, widely distributed applications, massively parallel applications, task farms, component composition frameworks, and more [12][13]. H2O simplifies security ensuring the safety of shared resources and that of users data via the well-established technologies like SSL, JSSE [14] for data transmission and JAAS for users authentication.

In Figure 1 is shown how H2O service components can be layered into different logical programming environments.

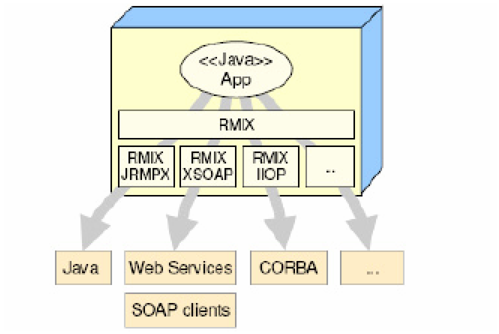


**Fig. 2.** The RMIX Architecture

## 2.2 RMIX

RMIX derives from the need to allow interaction among frame work adopting different communication protocols. Most solutions require protocol homogeneity at the endpoints, while one of the main goals of the H2O framework is to allow the dynamic enrolment of heterogeneous resources. The proposed solution is RMIX (whose name stands for RMI eXtended), a flexible framework for the unification of remote method invocation technologies [6].

The RMIX model allows selecting the underlying transportation protocol at runtime (see Figure 2). This live selection, based on Java dynamic linking capabilities, has to be supported by providing a jar file containing all the classes needed. The runtime selection capability allows removing any dependency from a specific protocol implementation and provides hot-plug extensibility.

The design requirements of RMIX are:

- The call semantics should be simple enough to be supported by simple and straightforward protocols;
- Nonetheless, the call semantics should allow the clients to take advantage of the full power of the H2O framework in a simple way;
- It should be possible to port existing protocols to RMIX without extensive modifications.

These requirements are not simple to be fulfilled. However, RMIX has taken advantage of the fact that Java RMI already provides a significant unification of the characteristics of most present RPC/RMI protocols. These unified characteristics are:

- The clients use stubs or proxies to access remote objects;
- The stubs or proxies implement the same interface that is exposed by the remote object;
- There is a base interface that provides the facilities needed to setup and tear down the communication path;
- The event of remote errors is signaled by some container exception;
- The parameter passing semantics are forced by the absence of a common address space.

These common characteristics form the core semantics of any RMIX call while the specifics of serialization and other features such as endpoint replication and distributed identity are delegated to the implementation of providers of single transport protocols [4].

## 3 The Web Services Gateway

As we described in previous section, H2O is a very powerful and extensible platform for the assembly of distributed application on top of a lightweight grid infrastructure and RMIX provides an extensible mechanism to access software components. However, in its standard distribution H2O lacks some fundamental features to enable integrating its dynamic capabilities into a full fledged Service Oriented Architecture. First, there is no monitoring facility; the framework provides the basic services to build a distributed monitoring system but none is readily available. Second, while RMIX is extensible and a SOAP provider is part of the standard distribution, there is no readily available service capable of automatically exposing H2O pluglets as Web Services.

In this section we will describe the components of the H2O Web Services Gateway and we will show how they fill the above mentioned gaps between H2O and a full fledged Service Oriented Architecture.

### 3.1 The RMIX Binding

The first component of the Web Services Gateway is a new WSDL binding dedicated to RMIX and the implementation of the Java classes needed to automatically generate the WSDL document from the Java pluglet interface. To achieve our goal we leveraged the WSDL4J [15] open source package and the org.w3c.dom Java package. The first step is the definition of a set of extension elements suitable to provide a complete description of RMIX while avoiding the adoption of an already made but restrictive one such as the one available for SOAP. These extensions will be realized as Java classes implementing the ExtensibilityElement interface defined by WSDL4J.

To define the extension elements it is necessary to associate a namespace to the RMIX binding. We chose to follow the standard format and we defined the namespace as shown below:

```
<definitions ....
xmlns:rmix=http://schemas.xmlsoap.org/wsdl/rmix/
... />
```

Obviously, at present the URL does not correspond to any actual schema. However, we chose to follow this convention to foster future standardization of the RMIX binding.

Next step consists of the actual implementation of the Java Classes.

A first group is composed by:

1. RmixAddress
2. RmixBinding
3. RmixOperation.

These classes have to implement the ExtensibilityElement interface in order to be part of a Definition object. Definition is the class that WSDL4J uses to objectify a complete WSDL document.

A second group is composed by:

4. RmixBindingSerializer
5. RmixBindingConstants.

RmixBindingSerializer implements both the ExtensionSerializer and the ExtensionDeserializer interfaces as defined by WSDL4J. This class provides the capability to marshal and un-marshal the ExtensibilityElements part of the previous group of classes.

Finally, RmixBindingConstants is a simple container class that collects all the constants needed to complete the definition of the RMIX binding.

The classes in the first group, namely RmixAddress, RmixBinding and RmixOperation, define the elements that plugs in the *binding* and *address* sections of the WSDL document. More in detail, the *binding* section is as follows:

```
<wsdl:binding name="RMIXBinding"
type="tns:NomePortType">

<rmix:binding style="transparent"/>

<format:typeMapping encoding="..." style="...">

<format:typeMap typeName="..." formatType="..." />

</format:typeMapping>

<wsdl:operation name="kernelInfo">

<rmix:operation

methodName="..."

methodType="..."

parameterOrder="..."

returnPart="..."/>

<wsdl:input name="...">

</wsdl:input>

<wsdl:output name="...">

</wsdl:output>

</wsdl:operation>

</wsdl:binding>
```

While the *address* section is as follows:

```
<wsdl:service>

<wsdl:port name="..." binding="tns:RMIXBinding">

<rmix:address plugletURI="..."/>

</wsdl:port>

</wsdl:service>
```

Besides the similarities that our mapping shows with the JavaObject binding as defined by WSIF, it is important to notice the presence of the *format:typeMapping* element. This element is not part of WSDL4J (although it is supported as an ExtensibilityElement). On the contrary it is part of the WSIF framework and it is used by

WSIF on the client side to allow direct mapping of Java types used by H2O to and from WSDL types, without an explicit and extensive *types* section of the binding.

In the *address* section it is important to notice the presence inside the *port* element of the ExtensibilityElement *rmix:address*. This element defines the attribute *plugletURI* that defines the invocation address of the H2O pluglet that the WSDL document exposes as a Web Service.

RmixBinding and RmixOperation are ExtensibilityElements of the *binding* section while, as mentioned above, RmixAddress is an ExtensibilityElement of the *address* section.

The structure of these classes is rather homogeneous. They all define the fields related to the WSDL document and the fields related to their interface with getter and setter methods. The class RmixOperation does not define a way to build the QName (qualified name) as a matter of fact the QName is generated automatically using the constants values defined in the RmixConstants class.

The RmixBindingSerializer class is used to marshal (and un-marshal) Java interfaces. Its structure has three main parts dedicated to three specific tasks:

1. marshalling
2. unmarshalling
3. registering the serializer.

This class leverages the standard WSDL4J DOMUtils class. More in details, the marshalling task makes heavy use of the printAttribute method. The actual operation starts analyzing the ExtensibilityElement to check if it is an instance of the RmixBinding class or of the RmixOperation class or of the RmixAddress class. Once the exact type has been identified, the code selects the specific attributes as defined by the different classes and serializes them by means of the PrintWriter associated with the Definition object.

The un-marshalling task reverses the operations described above and fills up the attributes according to the values contained in the document.

Finally, the registerSerializer task consists of the instantiation of a QName object corresponding to the element (e.g. *rmix:binding*), then of the declaration that the class *RmixBindingSerializer* provides marshalling and un-marsalling facilities for objects of the type identified by the QName.

The following Java code snippet shows the actual task implementation:

```
public void registerSerializer(ExtensionRegistry
registry) {

registry.registerSerializer(javax.wsdl.Binding.class,
RmixBindingConstants.Q_ELEM_RMIX_BINDING, this);

registry.registerDeserializer(javax.wsdl.Binding.class,
RmixBindingConstants.Q_ELEM_RMIX_BINDING, this);

registry.mapExtensionTypes(javax.wsdl.Binding.class,
RmixBindingConstants.Q_ELEM_RMIX_BINDING,
RmixBinding.class);
```

The code may be used to generate WSDL documents that contains elements of type rmix:binding, rmix:operation, rmix:address.

## 3.2 The Monitoring Pluglet

In the previous section we described the process used to generate a WSDL document capable of representing the interface exposed by an H2O software component (a pluglet) so that it may be automatically exported as a Web Service. In this section we show how we implemented a pluglet capable of monitoring the set of services deployed onto an H2O node. The deployment of a pluglet generates an H2O event that our pluglet is capable of capturing. Once the deployment has been detected, our monitoring pluglet extracts from the new service its URI (needed to set the value of the *rmix:address* attribute) and the path to the jar archive containing its code. The code of the new service is then analyzed using Java reflections to capture the functional interface of the new service so that newly deployed services can be automatically processed to produce their WSDL description.

The actual development of this monitoring pluglet requires three classes:

- The interface MonPluglet;
- The implementation MonPlugletImpl;
- A client to deploy, initialize and start the pluglet in an H2O kernel.

The default security policy enforced by H2O sandboxes each pluglet inside an environment independent from any other pluglet. In order to allow the monitoring pluglet collect information about other pluglets we had to set a special security policy for it. More in details we had to add to the Policy.xml file that defines the H2O kernel security policy the following lines:

```
<grant
codebase="${h2o.service.base.url}/prove/MonPluglet.jar">

<permission type="java.security.AllPermission"/>

</grant>
```

This grants to our pluglet the right to investigate the whole H2O kernel state instead of sanitizing it in its own sandbox.

The initialization of the monitoring pluglet instantiates two objects implementing the interfaces DeployListener and PlugletStateListener. These interfaces are defined by H2O that uses them to allow user implemented objects to receive notifications of internal kernel events.

The first interface defines a method:

```
plugletDeployed(DeployEvent evt)
```

This method is a callback that the kernel invokes to notify every object of type DeployListener that a new pluglet has been deployed inside the kernel; the monitoring pluglet uses this callback to capture the events of new services deployment. However, it is not sufficient to capture this kind of events. In fact, a newly deployed pluglet is not accessible as a service provider and it is not possible to connect to a pluglet until it has reached the lifecycle state ACTIVE.

Thus, it is necessary for the monitoring pluglet to be aware not only of deployment events, but also of changes in the other pluglets lifecycle state. We implemented this feature leveraging the other Listener interface defined by H2O, the PlugletStateListener.

**Table 1.** Java to WSDL elements correspondence

| **Pluglet** | **WSDL Definition** |
|---|---|
| Interface | portType |
| Method | Operation (rmixOperation, bindingOperation) |
| Input parameters | parameterOrder in rmixOperation |
| Return value | returnPart in rmixOperation |

This interface defines a method:

```
Changed(PlugletStateEvent evt)
```

This method is a callback; the kernel invokes it on every object implementing the PlugletStateListener interface every time a pluglet makes a transition in its lifecycle. The monitoring pluglet traces all the PlugletStateEvents: as soon as a pluglet reaches the ACTIVE state, it connects to this pluglet (the exact details of the connection to the pluglet are out of the scope of this paper) and analyzes it and its code to extract the information required to generate the WSDL document.

Once the connection to the deployed pluglet is made, we obtain the functional interface of the service offered by it by means of Java reflection. More in details, we use the correspondences we show in table 1.

Once the WSDL document has been generated according to the rules described in the previous section, the monitoring pluglet publishes it.

At present, the H2O Web Services Gateway neither implements a full fledged UDDI server nor is capable of interacting with an external one. The current solution for WSDL services description publication is limited to insertion in a jetty based web site. In future developments, we plan to extend the H2O Web Services Gateway with full UDDI compliance.

## 4 Conclusions

In this paper we have presented the H2O Web Services Gateway, a set of H2O software components capable of dynamically capturing the deployment of new software components into an H2O virtual machine and automatically generating and publishing the WSDL description of these components.

The H2O Web Services Gateway is composed by three main parts: *i)* the monitoring pluglet, *ii)* the Java2WSDL_Rmix service and *iii)* the WSDL publisher. Currently, there are two main limitations in our Web Services Gateway. First, the Java2WSDL_Rmix generator does not support the complete RMIX serialization semantics. Second, the WSDL publisher is a simple http server that allows clients to simply retrieve the WSDL description of the software components deployed inside an H2O virtual machine. In future versions we plan to extend the capabilities of the Java2WSDL_Rmix generator to achieve complete compatibility with RMIX. Furthermore, we plan to leverage the UDDI4J package to integrate inside the H2O Web Services Gateway the capability to interact with a full fledged UDDI server to achieve full compliance with Web Services standards.

Even with its current limitations, The H2O Web Service Gateway represents a significant extension to the H2O framework as it allows using standard Web Services client side invocation semantics and software, such as the Web Services Invocation Framework [16], to access software components deployed into an H2O virtual machine. A software developer can leverage the H2O Web Service Gateway to enable the automated export of software components deployed into an H2O virtual machine as Web Services. Thus, it is possible to easily integrate lightweight Grid application into Service Oriented Architectures.

## References

[1] Migliardi, M., Kurzyniec, D., Sunderam, V.: Standard Based Heterogeneous Metacomputing: The Design of HARNESS II. In: Proc. of the Heterogeneous Computing Workshop part of the International Parallel Distributed Processing Symposium 2002, Fort Lauderdale (FL), April 15-19 (2002)
[2] Sunderam, V., Kurzyniec, D.: Lightweight self-organizing frameworks for metacomputing. In: The 11th International Symposium on High Performance Distributed Computing, Edinburgh, Scotland (July 2002)
[3] H2O Project, home page, `http://dcl.mathcs.emory.edu/h2o/`
[4] Kurzyniec, D.: Towards lightweight and reconfigurable resource sharing frameworks, Ph.D. Thesis, Atlanta (GA), February 21 (2007)
[5] Sun Microsystems Inc., Java SE Security, `http://java.sun.com/javase/technologies/security/`
[6] Kurzyniec, D., Wrzosek, T., Sunderam, V.S., Slominski, A.: RMIX: A Multiprotocol RMI Framework for Java. In: Proc. of the International Parallel Distributed Processing Symposium 2003, Nice, France, April 22-26 (2003)
[7] AA.VV. Web Services Invocation Framework, `http://www.ws/apache.org/wsif`
[8] Migliardi, M., Sunderam, V.: The Harness metacomputing framework. In: Proceedings of the Ninth SIAM Conference on Parallel Processing for Scientific Computing, San Antonio (TX), USA, March 22-24 (1999)
[9] Migliardi, M., Sunderam, V.: Plug-ins, Layered Services and Behavioral Objects: Application Programming Styles in the Harness Metacomputing System. Future Generation Computer Systems 17(6), 795–811 (2001)
[10] Kurzyniec, D., Wrzosek, T., Drzewiecki, D., Sunderam, V.: Towards self-organizing distributed computing frameworks: The H2O approach. Parallel Processing Letters 13(2), 273–290 (2003)
[11] Sun Microsystems Inc., Java$^{TM}$ 2 Platform Enterprise Edition Specification, v1.4, `http://java.sun.com/j2ee/j2ee-1_4-fr-spec.pdf`
[12] Kurzyniec, D., Sunderam, V.: Combining FT-MPI with H2O: Fault-tolerant MPI across administrative boundaries. In: Proceedings of the 14th Heterogeneous Computing Workshop (2005)
[13] Kurzyniec, D., Hwang, P., Sunderam, V.: Failure resilient heterogeneous parallel computing across multidomain clusters. International Journal of High Performance Computing Applications (IJHPCA) (2005); Special Issue: Best Papers of EuroPVM/MPI 2004
[14] Sun Microsystems Inc., Java Secure Socket Extension (JSSE), `http://java.sun.com/products/jsse/`
[15] AA. VV. Web Services Description Language for Java, `http://sourceforge.net/projects/wsdl4j`

# A Flexible and Extensible Architecture for Device-Level Service Deployment

Thomas Frenken[1], Patrik Spiess[1], and Jürgen Anke[2]

[1] SAP Research CEC Karlsruhe, Vincenz-Prießnitz-Straße 1, Karlsruhe, Germany
mail@thomasfrenken.de, patrik.spiess@sap.com
[2] ubigrate GmbH
juergen.anke@ubigrate.com

**Abstract.** Integration of functionality and information from the Internet of Things (IoT) into the Internet of Services (IoS) is highly desirable but a complex endeavour. One hard to realize aspect is the remote deployment and configuration of services. While this has become commonplace in the business back-end, it is still a topic of research for networked embedded systems, mainly due to great heterogeneity. In this paper, we focus on remote management issues and propose a flexible and extensible architecture for systems performing deployment and configuration of services. The architecture is mainly targeted at environments comprising a large number of networked embedded devices, therefore integrating them effectively with the IoS. The key method of dealing with the observed heterogeneity is the dynamic exchange of algorithms (strategies) for steps of a common deployment process during runtime by configuration. Further on, we present an implementation of the architecture within the domain of future manufacturing.

**Keywords:** Service-Oriented Architecture (SOA), Internet of Things (IoT), Internet of Services (IoS), Service Deployment and Configuration, Deployment Planning and Execution, SOCRADES.

## 1 Introduction

Today, enterprises are moving more and more towards service-oriented architectures (SOA) composing applications and business processes by dynamically combining services. The application of the SOA paradigm to the whole internet leads to an Internet of Services (IoS). The key technology used for the realization of the services is web services (WS) based on the SOAP protocol. However, the IoS vision currently mainly includes services from the enterprise level. Another important trend, which is meant to shape the way business is handled in the future, is the Internet of Things (IoT). The IoT comprises millions of networked embedded devices also called smart items [1]. These devices are capable of collecting information about themselves, their environment, and associated devices and communicate this information to other devices and systems via the all-connecting internet. Many of them feature significant local computational

© Springer-Verlag Berlin Heidelberg 2008

power, allowing for dynamic configuration, installation, or execution of code. We summarize all these devices under the notion of smart devices.

The integration of information from the IoT into the IoS on the enterprise level has many advantages, e.g. be the replacement of manual keyboard entry with sensed location and state information of assets. Integration works efficiently if the smart devices expose their functionality as services (e.g. as WS). More powerful devices may host them internally, while for simpler devices encapsulation and abstraction through web service proxies is possible [2]. However, services will always be handled differently at enterprise and device level since device-level services are e.g. more fine-grained, focused on technology, and cannot guarantee high reliability. Additionally, systems at the device level are much more heterogeneous than at the enterprise level and lack universal communication standards.

In a dynamic service environment like the IoS remote deployment and configuration of services is essential. Although commonplace on the enterprise level, these functions are still a topic of research on the device level, where they are required for an effective integration of IoT and IoS.

The objective of the research project SOCRADES[1] is to develop an architecture for next-generation industrial automation systems. By integrating future manufacturing services with services from all corporate functions via SOA, more flexible processes with high information visibility will emerge. This includes WS hosted on smart devices within the IoT of manufacturing [3]. This work is part of a middleware developed in SOCRADES, which helps to overcome the differences between the service levels.

Within this paper we analyse the requirements for deployment and configuration of services for large populations of smart devices like the IoT and propose a high-level architecture for this purpose. The architecture is designed to be flexible enough for very heterogeneous environments and to be extensible to future scenarios and other application domains. It summarizes experiences gained during the implementation of deployment and configuration within the SOCRADES middleware, which has been used to perform these operations on simulated and physical devices in the scope of a set of future manufacturing scenarios. We describe implementation details, focussing on deployment planning and how we addressed the expected heterogeneity by following the strategy design pattern. The metrics used for selecting strategies are illustrated using a short sample scenario. A GUI, the Deployment Cockpit, is presented to supervise the deployment process and to visualize existing devices and services.

## 2 Related Work

Deployment and configuration of a distributed application's components is a challenging task even at the enterprise level. Deployment refers to the process

[1] `http://www.socrades.eu` (8 July 2008). The authors would like to thank the European Commission and the partners of the European IST FP6 project *Service-Oriented Cross-layer infRAstructure for Distributed smart Embedded devices* for their support.

between acquisition of software and its deprecation. Deployment planning, also called distribution planning or matchmaking, makes the decisions on how the software will be deployed onto the targeted distributed execution infrastructure [4]. The output of the deployment planning is a deployment plan whose execution leads to a certain deployment architecture. Configuration is the task of controlling and adapting a system's behaviour and deployment architecture to changed circumstances after it has been deployed.

Due to missing standards, many ad-hoc solutions for deployment and configuration of distributed applications have been developed. Frameworks like presented in [5] and [6] try to identify common functionality and standardize the development. A well known specification from the deployment domain is the OMG Deployment & Configuration (OMG D&C) [4] specification. It has been adopted and enhanced several times (e.g. in [5] and [7]). The architecture presented in this paper is more flexible than previous approaches since implementations for each step of the deployment process may be exchanged for each deployment job executed.

Deployment planning is an important operation within the deployment process and may influence a system's availability significantly. The general problem of deployment planning can be classified as a discrete, multi-objective, combinatorial optimization problem with constraints [8]. Therefore, finding a solution to the general deployment planning problem is most of the time NP-hard [9]. Several algorithms (e.g. [8],[9],[10],[11],[12],[13]) for deployment planning have been presented. Within this paper we present three algorithms for deployment planning which are adjustable to very heterogeneous deployment objectives.

Amongst others, information about target sites' capabilities and components' requirements is required for deployment planning. Available information is often very heterogeneous and sparse and many description formats are available. While in the past mostly key-value-pairs (properties) have been used for describing this information (e.g. [14]), there is a clear trend towards semantically enhanced description formats like the Ontology Web Language (OWL) organized according to various ontologies (like e.g. in [15] according to the FIPA Device Ontology[2]). While matching properties within the deployment planning is fast but inflexible, processing ontologies and reasoning over them is often more resource-consuming. It may not be expected that vendors will agree on using only one ontology [15]. The architecture presented within this paper is explicitly designed to be independent of any specific description format or ontology.

## 3 Requirements for Deployment and Configuration within the IoT

The following use cases have to be supported by a system for deployment and configuration. **Service Creation** is the first step within a common deployment process and includes implementing or composing a service as well as describing

[2] `http://www.fipa.org/specs/fipa00091/PC00091A.html` (10 July 2008)

additional metadata. **Service Publication** in modern SOAs is done within a registry repository and may involve validating services syntactically and semantically according to ruling guidelines (which are set up within a service governance process). **Service Updating** may become necessary for long-living services and might also include versioning of a service (or fragments of it). **Service Querying** allows to find published services and might be done using common query languages like SQL or XQuery, but also by semantic querying which involves reasoning over stored information. Decisions on where (i.e. into which execution containers) to deploy services may be done manually by a system architect or automatically in complex cases. **Service (Re)Mapping** decides automatically which services have to be available on which devices in order to execute certain business processes. Especially within the dynamic IoT, existing mappings may become disadvantageous or ill-suited (e.g. due to disappearing devices). In such cases existing deployment plans have to be (re)configured. **Service Deployment Execution** first translates general decisions of generated deployment plans into concrete actions to perform, and then executes these actions by sending suitable commands to targeted devices. **Service Monitoring** refers to collecting information about devices and service instances by either actively monitoring these or through receiving suitable events. The collected information is used during deployment planning and in order to detect possibly ill-suited deployment plans. **Service (De)Activation** can speed up deployment by deactivating currently unused service instances instead of uninstalling them. Later (re)activation may be faster than reinstallation and saves network traffic. Some additional functionality like Service Instance Discovery, Service Instance Querying, and Service Instance Selection is required in order to enable business process execution in dynamic environments. However, dynamic business process execution is out of the scope of this paper but is nevertheless supported partially by the architecture presented within this paper.

Additionally, a system for deployment and configuration within the IoT has to provide means to deal with a high degree of heterogeneity in three dimensions: **services**, **devices**, and **deployment objectives**. Services and devices are heterogeneous in many aspects especially because they are produced by a huge number of different vendors. Devices have different capabilities and are not equally capable of describing these. Available information is most of the time sparse and heterogeneous and is often delivered in different description formats. Devices capable of dynamically hosting services may have different deployment platforms, formats of deployable units, and communication protocols. Services may be described in different formats and different granularity. Deployable services may have implementations for multiple deployment platforms and may thus include arbitrary content (e.g. executable code, additional programming libraries, or sets of rules). The third dimension of heterogeneity are deployment objectives. A deployment job can either instruct the deployment system to compute a new deployment plan or to configure or revert an existing one. Further on, within each deployment objective category different requirements may be defined. For an initial deployment e.g. one might specify to either perform a dedicated deployment of a services to a given set of devices or to automatically generate a deployment plan listing only the services to map.

## 4 Proposed Architecture

The proposed high-level architecture for deployment and configuration systems contains components providing the required functionality while respecting the expected heterogeneity. Therefore, it is especially targeted to environments composed of large number of smart devices and is aligned to lessons learned during the implementation of deployment and configuration within the SOCRADES middleware. The main objective is to guide the design of systems for deployment and configuration and to enable an effective integration of the IoT into the IoS.

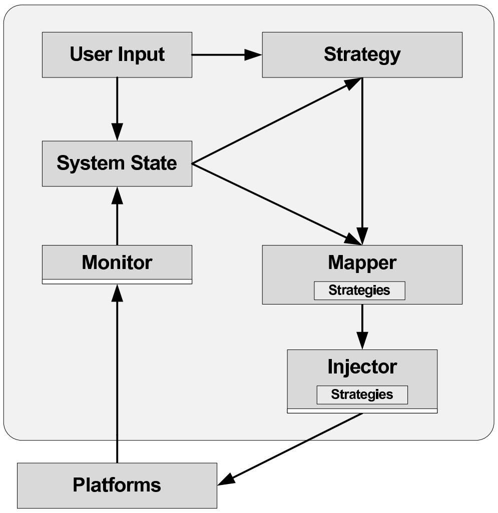


**Fig. 1.** High-Level Architecture for Deployment and Configuration

Figure 1 shows the proposed high-level architecture. Information required for deployment is delivered by (external system) users and by **Platforms** hosted on devices. The **User Input** component summarizes all information provided by (external) actors to the system e.g. by providing information about services or submitting deployment jobs. The **Monitor** monitors the nodes available within the Platform and provides information about devices and service instances deployed on those. In order to be able to deal with various existing platforms, each Monitor consists of a platform-independent part and a platform-dependent one. The latter hooks into the platform in order to retrieve information using platform-specific commands and protocols. Information provided by the Monitor and the User Input construct the **System State**, which contains, amongst others, all information about devices, interconnections, and services the architecture is aware of. The **Mapper** utilizes this information while performing deployment planning. The work process of the Mapper is guided by the **Strategy**, which decides about concrete implementations (strategies) to use for each step of the deployment process. The **Injector** is responsible for deployment execution which affects the Platform respectively the nodes within it. The Injector utilizes strategies to adapt its work process and consists very much like the Monitor of a platform-dependent and a platform-independent part.

The most important mean to deal with the expected heterogeneity are strategies. Strategies (as in design patterns) are exchangeable implementations for the most important steps performed within the deployment process. Which implementation to use for the steps of each single deployment job executed, is decided by the Strategy. The Strategy decides by considering user requirements (e.g. Quality-of-Service (QoS) criteria regarding the corresponding business processes) defined within deployment jobs, global system goals, and the System State. By using strategies, the architecture is able to address heterogeneity in the three dimensions mentioned in the last section.

## 5 Concrete Implementation

The implementation of deployment and configuration within the SOCRADES middleware was successfully used to realize a set of lab demonstrators for representative scenarios from the future manufacturing domain involving dynamic deployment and configuration of services on simulated as well as on physical devices. The implementation also contains a set of strategies for the various steps of the deployment process, a deployment platform, and a GUI (figure 3 (a)) for visualizing the system state as a graph and supervising the deployment process. The GUI may additionally be used to perform dynamic invocation of discovered service instances in order to simulate dynamic business process execution.

### 5.1 Implemented Strategies

The overall work flow of deployment planning and execution in SOCRADES is shown in figure 2. As already proposed in [10], we explicitly separated the deployment planning phase into two steps in order to be more flexible. Within the first step (node selection), suitable devices for services are selected, the second step (matchmaking) allocates services to devices. The input to the node selection step is the relevant system state, which is mainly comprised of all currently active devices, the topology of the systems (i.e. physical, network, and logical interconnects between devices), and the services to be mapped. The node selection strategies selected by the Strategy component interpret the given information (device capabilities and service requirements in one of an extensible list of description formats) and mark all devices with the IDs of services they are currently suitable for. This enriched system state is the input to the matchmaking step which then computes an assignment of $n$ services to $k$ devices that optimizes the goals defined while respecting given constraints. These constraints

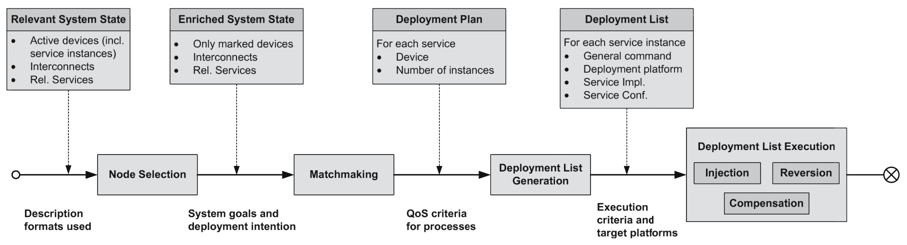


**Fig. 2.** Workflow Deployment Planning and Execution

are dynamic requirements of services upon devices and dependencies of services amongst each other (currently, description of dynamic constraints is limited to using properties). The output of the matchmaking step is a deployment plan that defines which services (and the number of these) shall be made available on which device.

Within this section we focus on the strategies implemented for the steps node selection and matchmaking. For the node selection step, currently two strategies are available, which leverage the advantages of different description formats. The first strategy is called **Property Expressions**. In order to utilize this strategy, device capabilities and service requirements have to be described as properties. For each requirement, services add an operator (e.g. *contains* or $\leq$) in front of their required value (e.g. a string or an integer). By combining the values of matching keys, an expression is formed which may either be evaluated to true or false. The Property Expressions strategy works very efficiently, but device and service providers need to agree on the keys to use a priori (or an additional matching list needs to be configured). As a second strategy for this step we implemented **Semantic Linking**, which utilizes ontological descriptions of devices and services written in OWL. During runtime, ontologies (created by vendors) describing devices and services are dynamically linked and a reasoner is used to proof a statement telling if the device is suitable or not. Although we provide a general base ontology, neither device nor service ontologies need to be based on it. The Semantic Linking strategy is extremely flexible in matching device capabilities and service requirements with the disadvantage of higher complexity in memory and time for linking the ontologies and performing the reasoning.

The first strategy available for the matchmaking step is called **Best Fit Decreasing Network** (BFDN). It is based on the idea of the Best Fit Decreasing (BFD) algorithm from the bin-packing domain. We modified the original algorithm in order to be suitable for deployment planning within the IoT e.g. by sorting devices and services according to separate, prioritized lists of properties and by exchangeable optimization goals. When matching n services to k devices, BFDN's worst case complexity is $O(n \cdot k)$, it is deterministic and neither complete nor optimal. The second strategy is called **Probabilistic Network** (ProbN). ProbN (algorithm 1) performs repeated search, starting each search with a random device within the search space and accepts that invalid solutions may be found. This Monte-Carlo based approach allows to search for solutions in different regions of the search space. The selection of devices is guided by a heuristic. First the device the last service was assigned to is chosen, next neighbouring devices are evaluated, and the last choice is a random device. ProbN may be configured to repeat the search up to a given number of iterations or to stop searching after a certain solution quality was reached. The objective function used to evaluate the quality of valid solutions is exchangeable by configuration. The implemented example objective function sums up the network hops between services, maximizing reliability of network connections and minimizing network bandwidth use. ProbN's worst case complexity is $O(n \cdot k)$, it is probabilistic and neither complete nor optimal.

**Algorithm 1.** Pseudo Code ProbN

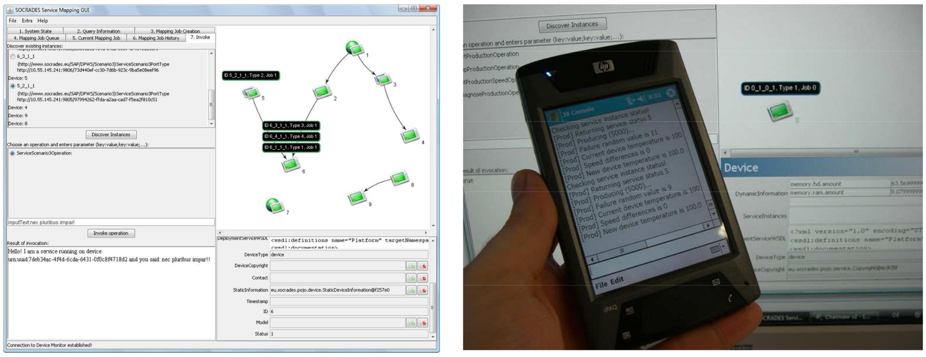
```
Require: device list $k$, service list $n$, service sorting criteria $serviceSort$, solution
    quality limit $l$, maximum iterations $m$

1.  sort $n$ according to $serviceSort$ descending
2.  $bestMapping$ = null
3.  while $m$ not reached AND quality of $bestMapping < l$ do
4.    $lastMapping$ = null
5.    for all $service$: $n$ do
6.      while $service$ not assigned do
7.        get next device $device$ according to heuristic
8.        if $device$ empty then
9.          next iteration
10.       if $device$ suitable then
11.         if capacity of $device$ not exceeded then
12.           if dependencies of $service$ satisfied then
13.             Update $lastMapping$ with assignment of $service$ to $device$
14.   evaluate $lastMapping$
15.   if $lastMapping$.quality < $bestMapping$.quality then
16.     $bestMapping = lastMapping$
17. return $bestMapping$
```

The last strategy for the matchmaking step is based on the implicit enumeration of solutions within the search space and is therefore called **Implicit Enum** (IE). It has exponential complexity of $O(n^k)$. However, IE may be configured to stop searching after any solution (with a minimum quality) was found instead of searching for the optimal solution. IE is the only strategy that guarantees to find a solution (provided that one exists) and will find the optimal solution, but is not suitable for many real-world situations including larger set-ups.

## 5.2 Basis for Decision Regarding Strategy Selection

The metrics (shown at the bottom of figure 2) used for selecting strategies for the process steps of the implemented deployment process heavily influence a system's flexibility. One objective of a more complex scenario from the future manufacturing domain is to monitor the average temperature of a certain area. Since we are focussing on the deployment planning part, we assume information about available devices and services has already been stored in a registry repository. The available devices are using different description formats (some use OWL, some properties). Our objective is to automatically deploy a temperature monitoring service on all smart devices which have a temperature sensor attached. The service instances have to be available as fast as possible. The deployed service instances are later on consumed by an enterprise-level service to monitor the average temperature.

Strategies for the node selection step are chosen according to the description formats used by the devices and services comprising the relevant system state. For each description format found, a suitable strategy is selected (if available) and enqueued in an execution queue. By executing this node selection strategy queue, there is no need to enforce the usage of a common description format or ontology. For the temperature monitoring example, both strategies, Property Expressions and Semantic Linking are executed. Matchmaking strategies are mainly selected according to the deployment's intention (manual/automatic mapping, configuration) and the overall objectives of the whole system (e.g. perform deployment as fast as possible or guarantee mappings have a quality above a certain threshold). For this example, BFDN is a good choice (automatic planning in reasonable time). Strategies for the deployment list generation step guarantee adherence to various Quality-of-Service (QoS) criteria (defined within deployment job descriptions) for the corresponding business processes (e.g. requirement of dedicated service instances). Selection of strategies for the deployment list execution step is done according to several execution criteria (e.g. (de)central, sequential or parallel execution with or without interconnection check between dependent services) and according to the deployment platforms of targeted devices. Implementing new strategies, as well as enhancing and standardizing metrics for strategy selection is part of future work.



Discover existing instances:
6_3_1_1
{http://www.socrades.eu/SAP/DPWS/Scenario3}ServiceScenario3PortType
http://10.55.145.241:9806/73d440ef-cc30-7d6b-923c-9ba5e08eef96
Device: 5
5_2_1_1
{http://www.socrades.eu/SAP/DPWS/Scenario3}ServiceScenario3PortType
http://10.55.145.241:9805/97994262-ffda-a2aa-cad7-f5ea2f810c51
Device: 4
Device: 9
Device: 8
Choose an operation and enters parameter (key:value;key:value;...):
ServiceScenario3Operation
inputText:nec pluribus impar!
Result of invocation:
Hello! I am a service running on device urn:uuid:7deb34ac-4f4d-6cda-6431-0f0c8f4718d2 and you said: nec pluribus impar!!
Connection to Device Monitor established!

| | |
|---|---|
| DeploymentServiceWSDL | `<wsdl:definitions name="Platform" targetNamespa` |
| DeviceType | device |
| DeviceCopyright | |
| Contact | |
| StaticInformation | eu.socrades.pojo.device.StaticDeviceInformation@f257e0 |
| Timestamp | |
| ID | 6 |
| Model | |
| Status | 1 |

```
Checking service instance status!
[Prod] Returning service status 5
[Prod] Producing (5000)...
[Prod] Failure random value is 11
[Prod] Current device temperature is 100
[Prod] Speed differences is 0
[Prod] New device temperature is 100.0
Checking service instance status!
[Prod] Returning service status 5
[Prod] Producing (5000)...
[Prod] Failure random value is 9
[Prod] Current device temperature is 100
[Prod] Speed differences is 0
[Prod] New device temperature is 100.0
```

**Fig. 3.** (a) Deployment Cockpit, (b) Deployment on Physical Devices

### 5.3 Integration of Physical Devices

To be able to fully control the deployment process, we have implemented a deployment platform (i.e. an execution container) that runs on devices hosting a Java VM supporting at least Java Mobile Edition 1.4 and the CDC profile. Additionally, we used a network stack that implements all WS-standards included in the Device Profile for Web Services[3] (DPWS) to expose hosted services as WS and in order to enable eventing etc. We successfully used this deployment platform to perform dynamic (un)installation, (de)activating, configuration, and monitoring of DPWS hosted services on simulated and physical devices. In particular we used the deployment platform on an HP iPAQ PDA (figure 3.b). Simulated as well as physically existing devices were integrated seamlessly into the deployment process and were visualized within the Deployment Cockpit.

## 6 Discussion

Three important attributes of a system for service deployment and configuration within the IoT are flexibility, extensibility, and scalability. Our high-level architecture is designed for flexibility in order to deal with the expected heterogeneity. Extensibility ensures that the architecture may be used in various domains and for future scenarios. Scalability is especially important in large set-ups.

Flexibility and extensibility are both enabled by using exchangeable strategies. Strategies for process steps collecting and comparing device and service information may be used to deal with heterogeneous and sparse information in various description formats. In case a new description format or deployment platform is meant to be supported, only a new strategy has to be implemented. Integration of this new strategy may be activated by configuration during runtime. Flexible selection of strategies also allows to deal with heterogeneous deployment objectives. By selecting different strategies for the matchmaking step, algorithms for allocation and configuration may be exchanged for each single deployment job, e.g. depending on the number of devices marked within the node selection step.

Scalability is especially important in the IoT (in both meanings: an Intranet or the Internet of Things) where large set-ups might contain millions of devices. In this case, the components of the architecture may be distributed. We have developed two distributed instances of the architecture. One guarantees central process control and adherence to global guidelines, the other does not. Component distribution is an advanced topic. Briefly said, in case central process control is required, components computing deployment decisions (Strategy, Mapper, and Injector) may only exist once and a consistent view on information relevant to deployment planning has to be guaranteed by the System State component. Components gathering information may be distributed, contributing information to the central System State. Without the need for central process control, all components of the middleware may be replicated, e.g. in several remote production sites.

[3] http://schemas.xmlsoap.org/ws/2006/02/devprof/ (8 July 2008)

## 7 Conclusion and Future Work

Within this paper we have presented a flexible and extensible high-level architecture for deployment and configuration of services. The architecture is especially targeted at large populations of smart devices hosting these services. By addressing the expected heterogeneity and specifics of such environments, the architecture is meant to integrate functionality of smart devices within the IoT into the IoS (represented by services on the enterprise level). The standardized communication pattern used is (web) services. The most important design pattern used for dealing with the expected heterogeneity are exchangeable implementations, so-called strategies, for all steps of a common deployment process. Strategies are automatically exchanged for each single deployment job according to certain metrics. The architecture was successfully applied within the domain of future manufacturing and was implemented within the SOCRADES middleware. The implementation contains a set of strategies for all common deployment process steps. In this paper we focused on strategies and metrics for deployment planning. Additionally, a deployment platform was implemented which was used for dynamic (un)installation, (de)activation, configuration, and monitoring of services on simulated and physical devices. A GUI, the Deployment Cockpit, visualizes the IoT and is used to supervise the deployment process.

The contribution of our paper is the analysis of requirements for deployment and configuration of services within the IoT, the design of a flexible and extensible high-level architecture, and an implementation of this architecture within the domain of future manufacturing in order to show the applicability of the architecture. The architecture is meant to transfer experiences gained to other deployment systems and domains. By enabling effective deployment and configuration of services on smart devices, the integration of functionality and information from the IoT into the IoS comes one step closer.

Future work includes the evaluation of the implementation within a prototypical manufacturing plant which is currently under construction. Results will be used to further enhance the implementation and design principles of the architecture. We are planning to integrate and evaluate more allocation and configuration algorithms proposed in research (like Avala [10] or PBFD [12]) as strategies. Further on, we will consider implementing support for standards like OMG D&C [4], CC/PP[4], and IUPP[5]. Additional deployment platforms like OSGi are already supported and will be tested. Semantic description of devices and services was already tested with promising results. However, scalability of semantic matching is a challenge, especially in larger set-ups of the IoT. We will also work towards more sophisticated ontology authoring for device and service descriptions. Finally, enhancing criteria and metrics for selection of strategies during deployment is an important part of future work.

[4] http://www.w3.org/TR/CCPP-struct-vocab2/ (10 July 2008)

[5] http://www.w3.org/Submission/InstallableUnit-PF/ (10 July 2008)

## References

1. Karnouskos, S., Spiess, P.: Towards enterprise applications using wireless sensor networks. In: 9th International Conference on Enterprise Information Systems (2007)
2. Anke, J., Müller, J., Spieß, P., Chaves, L.W.F.: A service-oriented middleware for integration and management of heterogeneous smart items environments. In: Proceedings of the 4th MiNEMA workshop in Sintra (2006)
3. de Souza, L.M.S., Spiess, P., Guinard, D., Köhler, M., Karnouskos, S., Savio, D.: SOCRADES: A Web Service based Shop Floor Integration Infrastructure. In: Floerkemeier, C., Langheinrich, M., Fleisch, E., Mattern, F., Sarma, S.E. (eds.) IOT 2008. LNCS, vol. 4952, pp. 50–67. Springer, Heidelberg (2008)
4. Object Management Group: Deployment and configuration of component-based distributed applications specifications (2007)
5. Shankaran, N., Balasubramanian, J., Schmidt, D., Biswas, G., Lardieri, P., Mulholland, E., Damiano, T.: A framework for (re)deploying components distributed real-time and embedded systems. In: Proceedings of the 2006 ACM symposium on Applied computing, Dept. of Electrical Engineering and Computer Science - Vanderbild University, Lockheed Martin Advanced Technology Labs, April 2006, pp. 23–27 (2006)
6. Malek, S., Mikic-Rakic, M., Medvidovic, N.: An extensible framework for autonomic analysis and improvement of distributed deployment architectures. In: Proceedings of the 1st ACM SIGSOFT workshop on Self-managed systems, pp. 95–99 (2004)
7. Deng, G., Schmidt, D.C., Gill, C.D., Wang, N.: QoS-enabled Component Middleware for Distributed Real-Time and Embedded Systems. In: Handbook of Real-Time And Embedded Systems, pp. 15.1 – 16.1. CRC Press, Boca Raton (2007)
8. Anke, J., Wolf, B., Hackenbroich, G., Kabitzsch, K.: Distributed applications and interoperable systems. In: 7th IFIP International Conference on Distributed Applications and Interoperable Systems (2007)
9. Hunt, G.C., Scott, M.L.: The coign automatic distributed partitioning system. In: Proceedings of the third symposium on Operating systems design and implementation, pp. 187–200 (1999)
10. Malek, S., Mikic-Rakic, M., Medvidovic, N.: Improving availability in large, distributed component-based systems via redeployment. In: Dearle, A., Eisenbach, S. (eds.) CD 2005. LNCS, vol. 3798, pp. 83–98. Springer, Heidelberg (2005)
11. Le Mouël, F., Ibrahim, N., Royon, Y., Frénot, S.: Semantic deployment of services in pervasive environments. In: Proceedings of the 1st International Workshop on Requirements and Solutions for Pervasive Software Infrastructures (2006)
12. de Niz, D., Rajkumar, R.: Partitioning bin-packing algorithms for distributed real-time systems. International Journal of Embedded Systems 2006 2, 196–206 (2006)
13. Zimmerova, B.: Component placement in distributed environment w.r.t. component interaction. In: Proceedings of the 2nd Doctoral Workshop on Mathematical and Engineering Methods in Computer Science, pp. 260–267 (2006)
14. Tryggeseth, E., Gulla, B., Conradi, R.: Modelling systems with variability using the proteus configuration language. In: The ICSE SCM-4 and SCM-5 Workshops, on Software Configuration Management, pp. 216–240 (1992)
15. Chrysoulas, C., Koumoutsos, G., Denazis, S., Thramboulidis, F.K., Koufopavlou, O.: Dynamic service deployment using an ontologybased description of devices and services. In: Proceedings of the Third International Conference on Networking and Services, p. 80 (2007)

# Fine-Grained Continuous Usage Control of Service Based Grids – The GridTrust Approach

Syed Naqvi[1], Philippe Massonet[1], Benjamin Aziz[2], Alvaro Arenas[2], Fabio Martinelli[3], Paolo Mori[3], Lorenzo Blasi[4], and Giovanni Cortese[5]

[1] Centre of Excellence in Information and Communication Technologies (CETIC), Belgium
{syed.naqvi,philippe.massonet}@cetic.be
[2] e-Science Centre, STFC Rutherford Appleton Laboratory, United Kingdom
{b.aziz,a.e.arenas}@rl.ac.uk
[3] CNR Institute of Informatics and Telematics, Italy
{fabio.martinelli,paolo.mori}@iit.cnr.it
[4] Hewlett Packard Italiana S.r.l., Italy
lorenzo.blasi@hp.com
[5] Interplay Software S.r.l., Italy
g.cortese@ipsoft.it

**Abstract.** Access control techniques designed for single domain infrastructures, where users are known by domain administrators, provide considerable liberty in the usage of resources. This paradigm is not suitable for highly scalable and decentralised systems such as Grids and service oriented architectures (SOA), where resources are shared between domains, and users come from remote domains. One approach is to provide policy-driven autonomic solutions that operate a continuous monitoring of the usage of resources by users. This paper presents the services and tools offered by the GridTrust Security Framework (GSF). GSF addresses three layers of the next generation of grid (NGG) architecture: the Grid application layer, the Grid service middleware layer, and the Grid foundation layer. The framework is composed of security and trust services and tools provided at the middleware and Grid foundation middleware layers. Various business case studies are being developed to validate the GridTrust results.

**Keywords:** Grid technology, usage control, service security, trust infrastructure.

## 1 Introduction

The *Service-based Grids* [Foster] have the ability to provide scalable and low cost service based infrastructures for both business and scientific purposes. They provide language- and platform-independent techniques for describing, discovering, invoking and orchestrating collections of distributed computational services, thus facilitating the development of complex wide-area applications [Gounaris]. However, from the security point of view, they introduce important challenges because the pool of resources and users are dynamic and managed by different administrative domains. Current access control technology in Grids only provides coarse grained security – i.e.

© Springer-Verlag Berlin Heidelberg 2008

user enjoys unlimited privilege of using the resource once he got access to it. The GridTrust consortium argues that coarse grained access control leaves Grids inherently vulnerable, and that not only the access to a resource needs to be controlled, but also the usage that is made of the resource. This paper presents the GridTrust framework that introduces fine grained and continuous usage control in Grids, and provides the necessary services, tools and methods to deploy it in service-based (OGSA compliant) Grids.

## 2 Current State of the Art in Grid Security

The native authorization system of Globus, the gridmap one, is too simple to satisfy the requirements of a cooperative distributed environment such as the Grid one. Hence, this section describes some attempts to enhance Globus security by integrating external authorization systems.

The Community Authorization Service, CAS, has been proposed by the Globus team [Pearlman]. CAS is a service that stores a database of VO policies that determine the actions that each Grid user can perform as member of the VO. A Grid user that wants to access a Grid service, requests to the CAS service a credential to access this service. The CAS returns a credential embedding a CAS policy assertion, and this credential is presented by the Grid user to the service he wants to exploit. This approach requires that Grid services are able to understand and enforce the policies embedded in the CAS credential, and these policies are coarse-grained, because they only define which of the local services can be accessed by the Grid user.

An alternative solution, described in [Thompson], integrates the Akenti authorization system in Globus. Akenti is an authorization system exploiting X.509 certificates for user identity and distributed digitally signed authorization policy certificates for access decisions. Once the user has been authenticated, the system retrieves the policies for each resource referred in the user request, and matches them with the user's credentials, that include the attributes assigned by the VO to that user. This solution is a pure pull model in which the user capabilities are collected after his authentication.

The solution presented in [Stell], instead, exploits PERMIS, which is a role-based access control infrastructure using X.509 certificates to define users' roles. All access control decisions are driven by an authorization policy that is stored in a X.509 certificate too. PERMIS supports classical hierarchical RBAC, in which roles are assigned to users and privileges on resources are paired with roles.

The Virtual Organization Membership Service (VOMS) [Alfieri] is another advanced authorization system for Globus. In VOMS a VO has a hierarchical structure with groups and subgroups; a user in a VO is characterized by a set of attributes, 3-tuples of the form *group, role, capability*. The combined values of all these 3-tuples form a unique attribute, the Fully Qualified Attribute Name (FQAN). A user contacts one or more VOMS server in order to obtain the authorization information granted by a VO to him. To access a Grid service the user creates a proxy certificate containing the information received from the VOMS Servers. To perform the authorization process the information is extracted from the user's proxy and combined with the local policy. The resource providers periodically query VOMS database to generate a list of VO users and map them to local accounts.

However, none of the previous systems performs fine-grained controls, i.e. controls the actions performed during the access. Moreover these models define static rights, because they depend on credentials that can be modified only by administrative actions, and these rights are evaluated only before granting the access, and no further controls are executed while the access is in progress.

Recently, Sandhu et al [Sandhu04] defined a conceptual model, called usage control (UCON), based on the concepts of mutable attributes and continuity of policy enforcement. In [Sandhu06] they propose the adoption of UCON in collaborative computing systems, such as the Grid. A preliminary attempt to adopt this model in Grid has been made in [Martinelli05], and this is the model that we adopt in the Grid-Trust framework [Martinelli07].

## 3 The GridTrust Framework

One of the outcomes of the GridTrust framework is the development of a set of online trust and security services and a set of policy modeling, analysis and transformation tools. The objective of developing these tools is to facilitate the development of rigorous Grid-based applications with enhanced security.

### 3.1 Virtual Organisation (VO) Model

In order to support rapid formation of VOs, we use the concept of virtual breeding environment (VBE) [Camarihna- Matos]. A VBE can be defined as an association of organisations adhering to common operating principles and infra-structure with the main objective of participating in potential VOs. We have adopted the view that organisations participating in a VO are selected from a VBE, as illustrated in Figure 1. Such organisations may provide resources/ services (ovals), and include users that utilise VO resources (small squares).

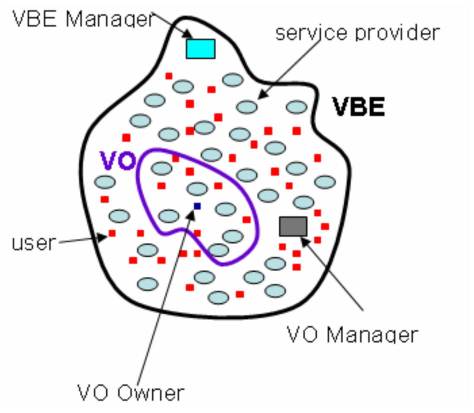


**Fig. 1.** Organisations and users in VBE

Organisations pre-register to a VBE via a VBE Manager component, including description of the resources they are willing to share in a Grid and the list of potential users belonging to the organisation. When a user requests to create a VO, s/he assumes the role of VO Owner and contacts a VO Manager with the description of needed resources. The VO Manager is in charge of selecting potential providers and setting up the VO to operation.

### 3.2 Framework Services

We describe here the different trust and security services that have been developed under the GridTrust framework.

- **The VBE Manager (VBEM)** has the main functionality of a service registry, where service providers register their services and other GSF services can retrieve them given abstract service descriptions. Each Virtual Organization (VO) is created within a specific Virtual Breeding Environment (VBE); and a VBE may contain several different VOs.
- **The VO Manager Service (VOM)** coordinates all the other security services and is the single point of access for users and service providers participating in the VO. The VO Manager is responsible for handling several functionalities. These include VO creation, populating VOs with services required by VO owners to achieve their goals, updating VO policies, evolving the VO by allowing its service providers to subcontract part of their services to other service providers and finally, terminating the VO.
- **The Policy and Profile Manager (PPM)** keeps all the knowledge bases needed by GSF services, namely: VBE and VOs users, with security preferences and their trust and reputation credentials; VOs with their owner and security policies; service providers with their services and the fine-grained security policies regulating access and usage of the services.
- **The Secure Resource Broker (SRB)** is called by VOM with a list of services, needed by the VO Owner to form its VO, and the associated security requirements. It returns the list of providers offering the requested services and also satisfying all the specified security requirements. One of those requirements is the reputation of a service in a VBE.
- **The Trust and Reputation Service (TR)** keeps track of the past and current behavior of VO owners, users and service providers and transforms it into trust and reputation credentials that can be considered by other users, service providers and GSF services when making decisions.
- **The Continuous Usage Control Service (C-UCON)** is an implementation of the UCON policy framework [Sandhu04], where it is deployed on each service provider and is responsible for the evaluation and runtime enforcement of policies about resource usage in VOs. It interacts with the TR service to get the current reputation of users and it also reports feedback to the TR service about users violating UCON policies.

Each of these services can be invoked only by mean of the API it exports, hiding all the implementation details on how the service is implemented. The framework is modular so it allows the possibility of adding future new security services if needed.

Figure 2 shows the different interactions among the GridTrust services when establishing and running a VO. This is done over several phases. In the first phase, a user (henceforth called the VO owner) requests from the VOM service the creation of a VO. In the next phase, the owner registers with the PPM service (through the VOM) the list of VO users and their security profiles. Then the owner requests from the

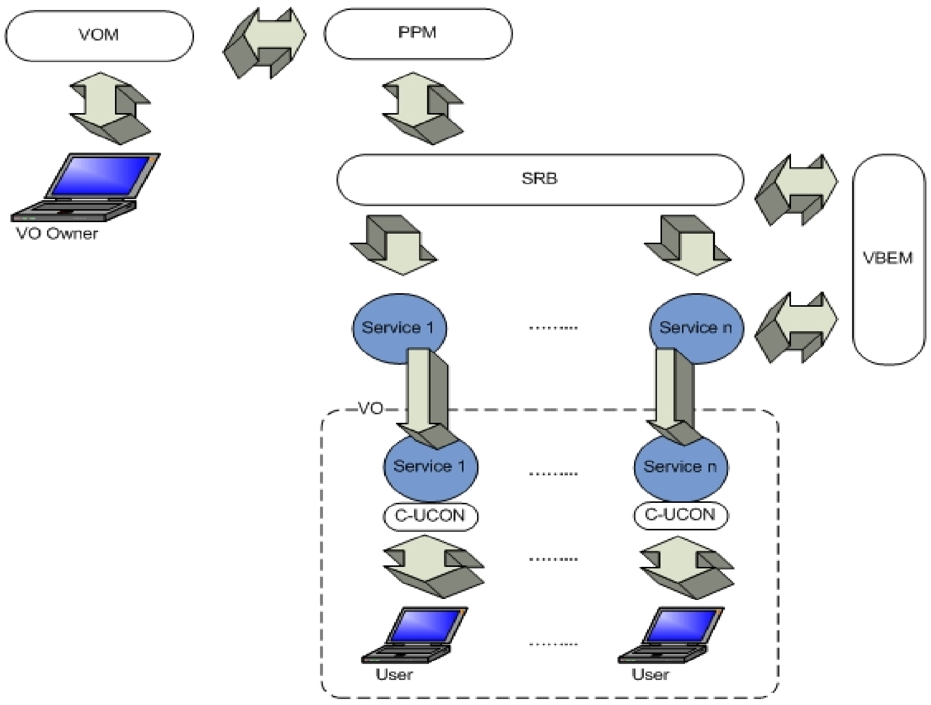


**Fig. 2.** Establishing a VO using GridTrust Services

VOM to search for suitable business services fulfilling its workflow by including the abstract description of each business service and any security requirements it must satisfy (e.g. minimum reputation level). The VOM utilizes the SRB service in its search and once the right candidates are reported back to the owner, the owner informs the VOM of its selection and SRB negotiates and schedules the selected candidates. The VO is now fully operational. Each of the computational resources underlying the business services is protected by a local instance of the C-UCON service, which monitors the users' behavior on those resources.

## 3.3 Policy Tools

These include a set of tools that are being developed to aid designers and analysts for policy writing during the design phases of the application development.

### 3.3.1 The Policy Requirements-to-Design Tool

The policy requirements-to-design tool facilitates linking of the security policies expressed in KAOS goal-oriented requirements model [vanLamsweerde] to the operational specifications of those policies. These policies are expressed in a formal process algebraic language POLPA [Martinelli06] [Martinelli07] [Aziz]. The tool eventually builds up a library of policies that comply with the specified UCON requirements.

#### 3.3.2 The Policy Refinement Tool

The aim behind the policy refinement tool is to allow policy designers to write VO-level policies using the stakeholders' alphabet and then refine it, in a correct and automatic manner, to a resource-level policy written using the resources' computational alphabet. The language at both levels is based on POLPA; however, the tool allows the designers to automatically derive low-level policies for VO resources. The tool also composes the refined VO-level policies with existing pre-VO policies at the resource level.

#### 3.3.3 The VO Modeling and Animation Tool

The VO modeling and animation tool generates Grid-based VO models in a formal refinement-based manner and then animates changes in the behavior of VOs when changes in the VO policies take place. We follow a formal approach that is based on the Event-B refinement language, where we envisage that the tool will develop interfaces for the Rodin modeling tool and the Pro-B animation tool.

### 3.4 Implementation Status

The first version of the GridTrust Framework has already been implemented and most of its source is available under the Apache 2.0 license. A public demonstration will be provided during the ICT 2008 conference in Lyon. To access further GridTrust publications and the current software release please refer to www.gridtrust.eu.

## 4 Validation Scenarios for GridTrust Framework

### 4.1 Distributed Content Management Case Study

The case study aims at researching and showcasing dynamic access and usage control mechanisms, similar to those outlined in [Sandhu06], for applications implemented in a Grid architecture.

The application outlined by this scenario is a *general purpose, workflow-enabled content management tool*, which supports a distributed organization in the execution of collaborative projects with the following characteristics:

- They aim at the production of some complex, sophisticated 'digital' product (e.g. a software system, or some multimedia product).
- They are 'knowledge-intensive' and 'content-intensive'. Workers depend on and need access to several sources of knowledge as well as digital content assets, which they assemble / use to create the product. This need must be supported by appropriate search facilities.
- The production process is structured along some workflow (e.g. a software production process, or a web / content publishing process), and foresees several phases. Policies which control access to these assets may vary according to the phase or state in the project workflow.

The application (a 'VO' in GridTrust terminology) offers access to a virtual content management ('CM VO') infrastructure, made out of several application servers, where users can: a) create a repository or collaborative ‘workspace’ where content can be stored b) upload content to such workspace c) search and retrieve content. Content managed through the infrastructure includes unstructured documents as well as multimedia content.

In VO Creation phase, the CM VO application discovers and registers application servers providing the actual CM services, thus creating the content management infrastructure.

In VO Usage phase, users access CM services while the GridTrust usage control infrastructure enforces appropriate access and usage control. The case study addresses two perspectives related to usage control: resource usage and collaboration. Overall, it aims at covering several of the types of usage control policies mentioned in [Sandhu06].

#### 4.1.1 Resource Usage

The CM VO allows on-demand provisioning of a content management infrastructure (See [Alfresco Cluster in the Cloud] for a similar scenario).

Users of the VO create a workspace, where they can store and share content with their partners, using the VO resources. The GridTrust infrastructure must ensure that users use VO resources in a fair and controlled way. To guarantee availability of resources and performance to all users of the CM VO, thresholds on the usage of resources must be enforced. For example:

- User can create and own at a given point in time in the CM VO only ‘max_spaces’ spaces and ‘max_content’ content items, occupying ‘max_disk_space’ i.e. the sum of the space requested to host the content objects owned by the user
- User can only perform a given number of queries in a time interval (e.g. in a minute/ hour/ day...). Queries can require non-trivial system resources, especially if they match a large number of content objects.
- Users can download only a given number of contents in a given time period.

Note that access/ usage control may be needed both at VO level and node level.

#### 4.1.2 Collaboration

The CM VO allows controlled sharing of such content resources among several users and organizations, which require traditional access control mechanisms. It also provides content workflow capabilities, hence should allow restricting or otherwise customizing the access to content / documents to specific users based on context. Types of policies we research in this case study include several dynamic, history-based access and usage control scenarios:

- Status of Shared Objects - access to granted based on the status of a content in a workflow
- Dynamic Separation of duties

Implementation of the application is work-in-progress. The CM VO is being implemented as a portal where VO users register and get services. Individual nodes providing content storage, indexing and query capabilities are implemented as Globus services interfacing to a JSR-170 Content Repository.

### 4.2 Supply Chain Case Study

The proposed Supply Chain scenario is based on two main ideas. The first is to use an auctioning system exploiting competition between transporters and allowing customers to find the best provider for each task. The second idea is to have route computing services, i.e. computational services providing maps and libraries to execute applications that solve the logistic optimization problem, to allow even SME transporters to optimize their routing. The routing computing service and possibly also the auctioning system are hosted on Grid resources.

Vigor, a pharmaceutical company, receives an order from a hospital. Vigor's warehouse has enough supply of the required goods, so only a transporter is needed to ship the order and satisfy the customer. Vigor's procurement system creates a Request for Quotation for the required transportation task specifying source, destination, expected arrival time, volume, weight and type of the goods and sends it to the Auctioning Service, thus creating an auction for it.

Celer, a transportation company, gets notified of the RfQ and its Automatic Quotation System analyzes the auction terms versus the company's policies and current availability of resources to determine if it's worth bidding or not. After a positive bidding decision the Automatic Quotation System needs to calculate the cost of delivering given the current resource engagement. For this calculation a job is sent for execution to the Grid Routing Service. The computation considers all the pending transportation tasks and time / capacity constraints for the Celer's fleet and optimizes (recalculates) the whole set of routes, one for each vehicle, to compute the incremental cost of executing the required transportation task. From the result an offer is created, with an estimated time for delivery, and a bid sent to the Auctioning Service. Choice of the best offer can be based on price, planned delivery time and transporter's reputation, depending on proponent's requirements.

To give a size to the scenario imagine a small group of 10 producers that create an auction for each of their 50 daily transportation tasks, and a group of 30 transporters that bid on every auction. Even for this small group it is 500 auctions per day (nearly

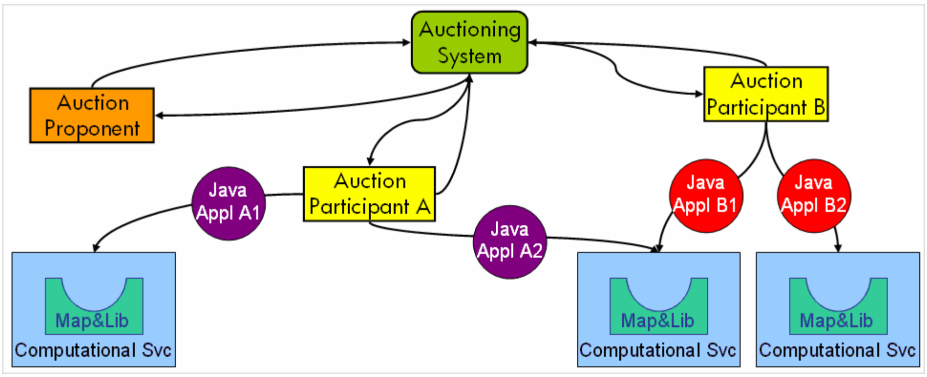


**Fig. 3.** Supply chain scenario's actors

one every minute in working hours), spawning 15.000 jobs of routing optimization every day. If the group of participant actors is not a small one the number of NP-complete problems to be solved in a single day may raise to several millions.

The components of the business scenario are the following (see Fig. 3):

- Auctioning system, a custom service running reverse auctions of type First-Price Sealed-Bid, allows producers to propose Requests for Quotations (RfQ))
- Auction Proponent, it's the Producer application (creates auction, receives result, creates Delivery VO)
- Auction Participant, is the *i*-th Transporter application (notified of an auction, creates Routing VO, invokes routing calculation, sends an auctioning offer)
- Map&Lib, are Routing support services, (maps, map access library, base routing functionality) made available by the Computational Service provider
- Java Appl, is the Routing application (executed on Computational Service) which may be different for each Auction Participant

The resources shared between domains are the Computational Routing Services, each hosting Distance-Time Matrices (maps) and providing sophisticated algorithms for solving the given Operating Research problem (e.g. VRPTW, see [Gambardella]).

This solution raises several security challenges such as selection of services with compatible security policies or continuous control of the execution of unknown applications, among others.

A future implementation of the system will allow monitoring the whole delivery phase and verifying transporter's compliance with the offered terms of service (considering the offer as a SLA). The reputation index of a transporter is based on a history of its accomplishments; with the current implementation it lowers if the transporter's behavior is not in line with VO security policies, but using a SLA monitor the reputation can be increased with successful shipments and lowered if the transporter doesn't fully comply with the terms agreed in the SLA.

## 4.3 Benefits of the GridTrust Framework

SRB is useful to clients as it allows to find Computing Services with compatible policies and granting access to the needed libraries.

UCON benefits Service providers in that continuously controls that the unknown code running on their servers is not violating policies or even executing harmful operations; UCON benefits clients too, because each executing application and data are protected against intrusion of other clients' applications.

TR is most useful to Service Providers by providing a measure of users' reputation, but may be useful to clients too when they want to base their partner's choice on the reputation index in addition to other parameters such as cost or performance.

VOM and VBEM at the end are the essential coordinators of the whole GridTrust Security infrastructure.

## 5 Discussions

Trust and security are fundamental issues in Grid, because of its collaborative and highly distributed nature. Grid users belong to distinct administrative domains that adopt distinct security mechanisms and have different security policies. These users are typically unknown, and no trust relations may exist among them. Hence, sharing resources in the Grid could be dangerous, because unknown and untrusted users could execute dangerous or even malicious applications on them. Another important issue is that accesses to services could be long-lived, i.e. could last hours or even days, and users' permissions may depend on conditions which are mutable over time.

The authorization systems adopted in Grid so far do not address all these issues. These authorization systems simply decide whether to allow a given user to access a service. No further controls are executed on the actions performed by the applications executed by remote Grid users on the local resource. Otherwise in GridTrust both VO Owners and Service Providers can define fine-grained security policies which are then continuously enforced by the GridTrust C-UCON Service.

The GridTrust project is innovative because it addresses the main security issues of Grid environments by proposing an integrated framework that provides a set of services performing the main Grid interactions in a secure way. These tools allow the Grid participant to create and manage VOs, to select resource providers having certain security requirements, to manage users' reputation, and to execute applications on behalf of remote Grid users while performing a fine-grained and continuous monitoring of computational services according to the UCON model.

Hence, a main advantage of the GridTrust framework is that it is not a simple authorization system, but consists of a set of services enhancing the security of the whole Grid lifecycle, from VO formation to VO dissolution.

Another interesting feature of the GridTrust framework is that all the components have been developed as Globus services and have been integrated in the Globus environment. Hence, the GridTrust framework could be adopted in current Grid nodes built on Globus with minor modifications. For the same reason, the GridTrust components could be easily integrated with other Globus based (security) services.

## 6 Conclusions and Perspectives

The GridTrust framework addresses the security and trust requirements of service-based Grids. In this paper, we have presented its approach for fine-grained continuous usage control of Grid resources. The Continuous Usage Control service of the GridTrust framework controls the usage of Grid's computational resources by applying fine-grained and history-based access control, and improves state of the art with mutable attributes, obligations and continuous enforcement. The GridTrust framework features fine grained monitoring of the actions performed by applications on the resources. The history of these actions is used in the evaluation of new requests. The access rights are therefore dynamic in GridTrust framework because attributes and conditions may change over time. The two presented use cases confirm GridTrust framework's workability.

## Acknowledgements

The research leading to the results presented in this paper has received funding from the European Union's sixth framework programme under grant agreement number FP6-033817.

## References

[Alfieri] Alfieri, R., Cecchini, R., Ciaschini, V., dell Agnello, L., Frohner, A., Gianoli, A., Lorentey, K., Spataro, F.: VOMS: An Authorisation System for Virtual Organizations. In: Proceedings of 1st European Across Grid Conference (2003)

[Alfresco Cluster in the Cloud] Alfresco Cluster in the Cloud, `http://ihatecubicle.blogspot.com/2008/05/alfresco-cluster-in-compute-cloud.html`

[Aziz] Aziz, B., Arenas, A., Martinelli, F., Matteucci, I., Mori, P.: Controlling Usage in Business Processes Workflows through Fine-Grained Security Policies. In: Furnell, S.M., Katsikas, S.K., Lioy, A. (eds.) TrustBus 2008. LNCS, vol. 5185. Springer, Heidelberg (2008)

[Camarihna-Matos] Camarinha-Matos, L.M., Afsarmanesh, H.: Elements of a base VE infrastructure. Journal of Computers in Industry 51(2), 139–163 (2003), `http://www.uninova.pt/~cam/ev/CiI.PDF`

[Foster] Foster, I., Kesselman, C., Nick, J., Tuecke, S.: Grid Services for Distributed System Integration. IEEE Computer 35(6), 37–46 (2002)

[Gambardella] Gambardella, L.M., Taillard, E., Agazzi, G.: MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows. In: Corne, D., Dorigo, M., Glover, F. (eds.) New Ideas in Optimization, pp. 63–76. McGraw-Hill, New York (1999)

[Gounaris] Gounaris, A., Paton, N., Sakellariou, R., Fernandes, A., Smith, J., Watson, P.: Modular Adaptive Query Processing for Service-Based Grids CoreGRID Technical Report TR-0076 (March 2007)

[Martinelli05] Martinelli, F., Mori, P., Vaccarelli, A.: Towards Continuous Usage Control on Grid Computational Services. In: Proceedings of Joint International Conference on Autonomic and Autonomous Systems and International Conference on Networking and Services (ICAS-ICNS 2005), p. 82. IEEE Computer Society, Los Alamitos (2005)

[Martinelli06] Martinelli, F., Mori, P., Vaccarelli, A.: Fine Grained Access Control for Computational Services. Technical Report Number TR-06/2006, Istituto di Informatica e Telematica, Consiglio Nazionale delle Ricerche, Pisa (2006)

[Martinelli07] Martinelli, F., Mori, P.: A Model for Usage Control in GRID Systems. In: Proceedings of the First International Workshop on Security, Trust and Privacy in Grid Systems (GRID-STP 2007) (2007)

[Pearlman] Pearlman, L., Kesselman, C., Welch, V., Foster, I., Tuecke, S.: The Community Authorization Service: Status and Future. In: Proceedings of Computing in High Energy and Nuclear Physics (CHEP 2003) (2003)

[Sandhu04] Sandhu, R., Park, J.: The UCONABC usage control model. ACM Transactions on Information and System Security 7(1), 128–174 (2004)

[Sandhu06] Zhang, X., Nakae, M., Covington, M.J., Sandhu, R.: A usage-based authorization framework for collaborative computing systems. In: Proceedings of the Eleventh ACM Symposium on Access Control Models and Technologies. SACMAT 2006, Lake Tahoe, California, USA, June 07 - 09, 2006, pp. 180–189. ACM, New York (2006)

[Stell] Stell, A.J., Sinnott, R.O., Watt, J.P.: Comparison of Advanced Authorisation Infrastructures for Grid Computing. In: Proceedings of High Performance Computing System and Applications, HPCS 2005, pp. 195–201 (2005)

[Thompson] Thompson, M.R., Essiari, A., Keahey, K., Welch, V., Lang, S., Liu, B.: Fine-Grained Authorization for job and resource management using Akenti and the Globus toolkit. In: Proceedings of Computing in High Energy and Nuclear Physics (CHEP 2003) (2003)

[vanLamsweerde] van weerde, L.: Requirements Engineering in the Year 2000: A Research Perspective. In: Proceedings of the 22nd International Conference on Software Engineering, Limerick, Ireland, pp. 5–19. ACM, New York (2000), http://www.sis.uncc.edu/~seoklee/teaching/Papers/lamsweerde00requirements.pdf

# An Approach to Identity Management for Service Centric Systems

Laurent Bussard[1], Elisabetta Di Nitto[2], Anna Nano[1], Olivier Nano[1], and Gianluca Ripa[3]

[1] European Microsoft Innovation Center, Aachen, Germany
[2] Politecnico di Milano, Milan, Italy
[3] CEFRIEL, Milan, Italy
{LBussard,AnnaW,ONano}@microsoft.com, dinitto@elet.polimi.it, gianluca.ripa@cefriel.it

**Abstract.** Today users consume applications composed by services from different providers across trust domains. By experience we know that security requirements and user identity management make services composition difficult. We believe that delegation of access rights across trust domains will become an essential mechanism in services composition scenarios. Users care about security but cannot deal with the variety of existing solutions for access control. A unified interface of access control and delegation is essential for multi-domain composite services. This paper addresses the problem of identity management for service-centric systems and proposes a novel approach based on an abstract delegation framework supporting different access control mechanisms. We show how the abstract delegation framework is designed to give control and clarity to the user consuming applications based on service composition. Besides the theoretical aspects, the paper shares experiences based on scenarios from the automotive industry.

## 1 Introduction

Agility and dynamicity are the focus of today's businesses. Service centric systems enable companies to react quickly to new opportunities. A service centric system is an application composed by services from different providers across different trust domains. Users access service centric systems which, in turn, access services that might be in different trust domains.

In a multi-trust domain environment accessing and invoking services requires proper authentication and access rights. With state of the art access control a proper federation of all the services is established before hand, the user will have to give his identity and credentials to enable the services to be invoked.

From the perspective of identity management (IdM), service centric systems are different from any other software system in the fact that they are built by composing existing services that are not necessarily under the control of the owner of the service centric system. This means that composed services may have access policies that are different from those of the service-centric systems that are exploiting them [1], and potentially belong to different trust domains.

© Springer-Verlag Berlin Heidelberg 2008

Various initiatives exist to address the issues mentioned above (we describe some of them in Section 7) but most of them are only suited for single trust domains or impose a specific authentication mechanism.

In this paper we are particularly interested in service centric systems which allow dynamic replacement of services. For example, if the service centric system needs to access the user's calendar on line, the actual calendar being accessed (e.g., Google Calendar, Microsoft Exchange ...) will depend on the preference of the specific user. If a new calendar service enters the market the service centric system will have to adapt at runtime to take advantage of this new service. Moreover, either the service centric system is in a federation of trusted domains with all the possible calendar services, or we should find a way for the user to dynamically and explicitly delegate his access rights to the service centric system on his specific calendar. In this paper we investigate this second possibility by developing a proper delegation framework and integrating it into SCENE, an existing platform that supports the development and execution of self-adaptable service centric systems. Our work is particularly useful when a service centric system needs to access a service on behalf of its user, e.g., for acquiring some sensible information.

The rest of this paper is structured as follows. In Section 2 we present the main issues of IdM for service-centric systems. In Section 3 we describe the main ideas behind our approach. In Section 4 we shortly present SCENE, that is the runtime infrastructure we have built within the SeCSE project [2] to support the execution of adaptable service-centric systems [3]. In Section 5 we show how our approach to IdM has been integrated within SCENE. In Section 6 we present a case study from the automotive domain, and in Section 7 we present some related work. Finally, in Section 8 we draw the conclusions and highlight the challenges for future work.

## 2 Service Centric Systems and Identity Management

As we mentioned in the introduction we are interested in service centric systems which enable to dynamically select and adjust which services to use. There are some approaches in the literature that enable this (see [4] for a classification of these) and we adopt SCENE. As we will discuss in Section 4, in SCENE a service centric system is defined in terms of two main aspects: the application-dependent control logic that is expressed using BPEL, and the policies that enable self-adaptation at runtime. These policies, among other things, allow designers to postpone the binding to specific component services until runtime. At design time the designer associates to a BPEL service invocation an *abstract service*, e.g., a generic calendar service in the example mentioned in the introduction, and defines a binding rule that defines how to determine the binding to a concrete service. In our specific example, this rule states the dependency between the user's profile and the actual calendar to be used.

As mentioned before, composition of services becomes more complicated when IdM has to be taken into account and the choice of a concrete service for composition depends on the identity of the user. Moreover, when services contain personal information (*personal services*), the user wants to explicitly control who can access them.

Indeed, in the case of service centric systems different parts of the composition can be owned and operated by different parties, possibly from different trust domains. Therefore the relation between services in a service composition is more dynamic and involves trust domains that are not federated a priori.

As such, the challenge is twofold. First, to enable composition of services from different trust domains potentially protected by different types of access control mechanisms. Second, at the same time, putting the user in control of who can access his personal services and letting him make the choice if he wants to delegate access rights to a service.

Our solution to the problems mentioned above is to build an abstraction layer that takes care of delegation by addressing the following requirements:

– *User Control*: Users generally want to keep control on their assets including personal services (e.g., calendars, online shared pictures, or medical records). Unfortunately, the different access control mechanisms and management interfaces in place make it very difficult to delegate rights and to keep an overview on who can access what. This is even worse when composite services start asking for access rights. The abstraction layer should define a unified user experience on top of various access control mechanisms.
– *Separation of concerns*: When multiple services from different trust domains have to be combined, details regarding access control and management of access control should not “pollute” the composition. The abstraction layer should isolate the composition from delegation details.
– *Dynamicity*: Personal services, for their nature, cannot be bound to a composition at design time, unless that composition is built only for a specific subset of users. For instance, a composite service requiring the real-time location of its user needs to know which location service this user is registered to and how to authenticate to this service. The abstraction layer should support dynamic binding and should resolve the delegation issues at run time.

## 3 Overview of Abstract Delegation

To address the points above, the delegation framework offers an abstraction layer for delegation as well as a mechanism to map abstract delegation to concrete mechanisms. In this paper, authors refer as ‘delegation’ any form of delegation of rights.

The abstraction covers the following three types of actors:

– The *Delegator* is the party that grants some rights regarding its personal services. In many scenarios, the user of a composite service acts as a delegator.

- The *Delegatee* is the party that requests to be granted access. The delegatee gets the address of the personal service to be used and a credential to authenticate to this personal service. The type of credential varies from service to service (user-name/password, X.509 certificate, SAML token, etc.).
- The *Resource* is the personal service chosen by the delegator and accessed by the delegatee.

Using the delegation framework, the delegator is thus able to specify (in a composition or using a unified UI) that an abstract delegatee has some access rights on an abstract resource. This abstract delegation is mapped at runtime to a concrete mechanism. For instance, the XACML policy controlling access to the resource could be modified or a new SecPAL credential could be created and handled to the delegatee. Abstract revocation of access rights is also mapped to appropriate mechanisms, e.g. modification of a policy, or adding an entry in a revocation list. The mapping is done based on a system of plug-ins as described in more details in [5], [6] and [7]. Authors do not consider impersonation as a form of delegation because the mains goal of this work is to enable users to control who can access his resources. Impersonation, i.e. handing one's credential to another party, excludes user control.

Figure 1 gives an overview of delegation: a delegator owns a resource and gives some privileges to a delegatee to access this resource. For instance, Bob (delegator) is registered to an on-line calendar (resource) and gives read access to Alice (delegatee).

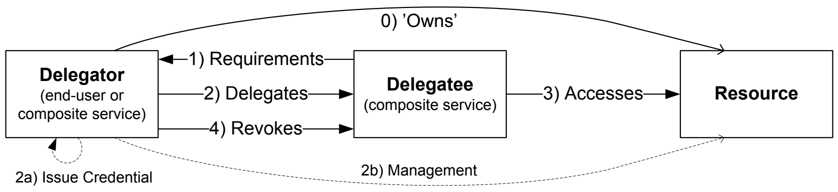


**Fig. 1.** Overview of delegation

Authors do not assume that delegators always host resources. However, delegators must have a way to specify who is authorized to access their resources. Common examples of resources are on-line medical records, calendars, location services, and photo galleries. The resource guard that deals with access control decisions does not have to be part of the resource and can typically be a Security Token Service (STS) [8] or a policy decision point (PDP) [9].

Delegation results in creating a credential that is provided to the delegatee (step 2a in Fig. 1), modifying resource-side policy (step 2b in Fig. 1), or a combination of both.

## 4 Overview of SCENE

The SCENE platform provides the runtime execution environment for compositions written in the SCENE language. As mentioned in Section 2, the SCENE language extends the standard BPEL language with rules that are used to guide the execution of self-adaptation operations at runtime. Figure 2 shows an example of a rule, written in pseudocode, referred to the example of the calendar we have introduced before. The rule allows the runtime environment to bind to the calendar that has been set in the user's profile. Of course, different calendars could have different WSDL interfaces. The runtime platform takes care of such differences by adopting proper adapters as explained in [10].

The SCENE prototype includes the following components:

– a BPEL engine, Active BPEL [11], which is in charge of executing the process part of the service composition;
– an open source rule engine, Drools [12], responsible for running the rules;
– WSBinder [13], responsible for executing binding actions at runtime based on the directions defined in the rule language;
– a set of Proxies that decouple the BPEL process and the execution environment from the logic needed to support reconfiguration.

The components of SCENE interact through a publish-subscribe paradigm. At runtime, when the execution of the process reaches the invocation of an external service, a proxy operation is actually called. If the proxy does not refer to any concrete service, it emits a `bindingEvent`. The rule engine – that has subscribed to this event – receives it and activates a rule able to handle – possibly with the activation of WSBinder – the missing binding. The control is then passed to the proxy that, possibly activating an adapter, invokes the proper operation on the bound service, and then passes the control back to the BPEL execution environment.

```
Event: bindingEvent
Condition: action=checkSchedule
   userProfile.Calendar not empty
Action: bind checkSchedule to userProfile.Calendar.checkIfBusy
```

**Fig. 2.** Rule that establishes the binding with a specific calendar

## 5 Solution Architecture: SCENE and Abstract Delegation

In this section we present the architecture of the IdM solution as it has been incorporated into SCENE. Figure 3 shows the components of SCENE that are relevant to the integration, i.e., the BPEL engine and a proxy, and the components of the delegation approach. In the figure, the client represents the user interface of a service centric system. Such client must have a callback interface for allowing the delegation framework to ask for delegation. Also, it incorporates a delegation browser that keeps track of all delegations that have been given by the user. The figure highlights a service that is the one being called at a certain time. It exposes a *management interface* in order to allow the management of its Access Control Lists. The registry stores the specifications of all services that, from the perspective of this paper, include the WSDL interfaces and policies.

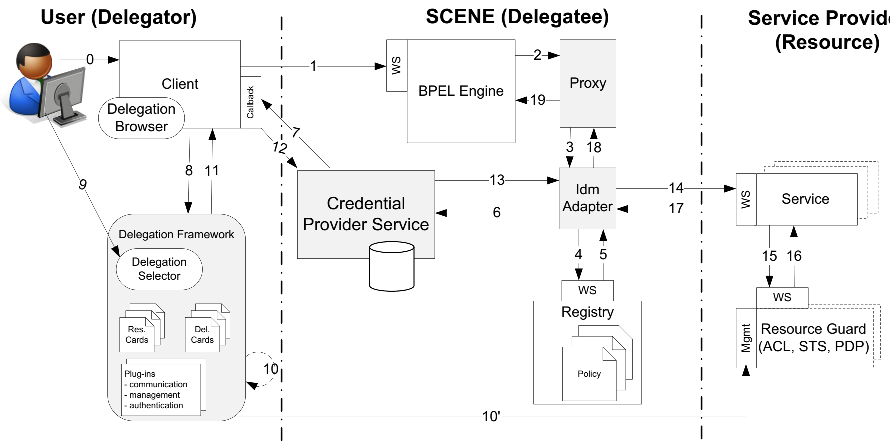


**Fig. 3.** Integration of SCENE and the Delegation Approach

The main components of the Delegation Approach are:

- A *Resource Card* contains the information necessary to delegate access. Resource cards should typically be provided by service providers. For instance, when registering to a medical record service, the user should get a resource card specifying the delegation model, the address of the management interface, the credential to use to authenticate, and the granularity of the privileges. The abstract privilege is a higher representation of the underlying privileges that can be delegated.
- A *Delegatee Card* identifies a delegatee. The delegatee card of a mash-up could contain its X.509 certificate while the delegatee card of Alice could contain her fingerprint and/or her corporate alias.
- The *Delegation Framework* that is the core of the system and manages the cards owned by the user. As mentioned in Section 3, the cards are related to resources, i.e., the personal services, and to delegatees. The Delegation Framework is installed at the Client site and assumes that the delegator authenticates before using it. More details on this component are provided below.
- The *Credential Provider Service* that is the server-side component of the delegation framework. It is invoked by SCENE as a web service and enables delegation for the service centric system.

– The *IdM Adapter* that is part of the proxy. It retrieves from the registry the policy of the service to be invoked, contacts, when delegation is needed, the credential provider service, and constructs the invocation message to be sent to the service based on the output provided by the credential provider service and on the policy published by the service.

The delegation framework offers three main functions: `Delegate` is used to delegate rights. This can be achieved by creating a chain of credentials with, for instance, SPKI [14] or SecPAL [15]. Another form of delegation results from the modification of an authorization policy, e.g. XACML [9], or from adding an identity to an access control list (ACL). `Revoke` makes it possible to revoke rights. Depending on the underlying delegation models, revocation may imply modification of a policy or adding an entry to a certificate revocation list. Finally, `GetDelegationStatus`, returns a list of ongoing delegations (`DelegateeCards` and `AbstractPrivileges`) that is used to keep an overview of the delegation status.

A plug-in mechanism is used in the Delegation Framework to support different delegation models. The following plug-ins have been implemented to validate the framework, but of course others could be easily implemented:

– *SecPAL Plugin:* It is based on the resource's SecPAL policy and the delegator's SecPAL credential. It creates a new credential stating that the delegatee can access a subset of the resource for the required duration.
– *STS Plug-in:* It is developed to show that also legacy security token services (STS), BizTalk Services Identity Provider[4], can be managed by the Delegation Framework.
– *Google Calendar Plug-in:* It is used when the Delegator needs to grant access to the Google Calendar. It shows the feasibility of interacting with non-SOAP legacy management interfaces as the ones of Google that offers a REST API[5].
– *Fingerprint Plug-in:* This plug-in is not related to service composition but has been developed to look at resources that are neither SOAP web services nor REST. This plug-in has been used to control access to a car.

In Section 6 we clarify the role of all components through an example.

## 6 A Case Study

We are currently evaluating our approach in some service centric systems that are being developed by the industrial partners of the SeCSE project. Here we focus on one example that is taken from the automotive domain. This example is a composed service called `XTRIP` that helps drivers of some cars in keeping their schedule updated depending on the status of their travel. The service allows a driver to plan a trip. Based on the plan and on a navigation system that allows the service to know the geographical position of the car, the service itself

[4] See `http://labs.biztalk.net/identity.aspx`
[5] See `http://code.google.com/apis/calendar/`

is able to automatically check the calendar of the driver to make sure that he will be on time for the scheduled appointments. In case of problems, the service automatically establishes a telephone communication between the end user in the car and his secretary so that they can take actions to change the schedule as needed. XTRIP exploits some component services to collect data about the current position of the car, to access the user's calendar, and to establish a telephone call when needed.

This example highlights the two aspects that are relevant in this paper, that is, dynamicity of bindings and need for some IdM mechanism.

As for the first issue, besides the selection of the concrete calendar service that should depend on the preferences of the actual user which requests the service, other cases of dynamic binding are highlighted. In particular, the selection of the actual service for planning the trip could be decided at runtime based on the geographical location of the user. This way, it is possible to take advantage of navigation systems specialized on specific areas. Also, the selection of the service to establish the telephone call could depend on the telecom provider that offers the best rate to connect the traveler with his secretary and also on performance. To address these issues we exploit the rule-based approach offered by SCENE as shortly explained in Section 4 and detailed in [3].

As for the second issue, different IdM needs are coexisting here. As the reader can imagine, both the calendar and the phone call services are likely to require some access control and delegation mechanism. In the case of the phone call service we can imagine that the provider of the composite service can preliminarily establish agreements with some telecom providers that, in turn, grant access to the composite service according to a pay per use policy. In principle, this interaction between XTRIP and the telecom service does not require any sensible information about the XTRIP user to be passed to the service, with the exception of the phone numbers to call that XTRIP will be able to pass to the telecom service without the need of any explicit reference to its owners. Thus, in this case the access to the telephone service does not require any awareness and delegation by users.

In the case of the calendar, on the other hand, the identity of the XTRIP user is very relevant. Without knowing it, the calendar would not been able to provide to XTRIP the information about the user's appointments. Indeed, the calendar should be able to check that the user has granted access to his information to XTRIP. Only by presenting some evidence of the delegated rights to the calendar service XTRIP will obtain the user's appointments. Moreover, different calendars could be exploited by differend users, and they could have different delegation models and access control mechanisms.

So far, we have experimented with the approach of using two different calendar systems that have been developed independently from each other and from our platform. The first one is the Google Calendar service [16]. It allows users to create one or more calendars associated with an account and to share them with other users (with various access rights). It is not a web service but it provides a REST API that we have used to wrap the service as a web service. This service has also associated a Resource Card. This card states that delegating the access rights to the Calendar is accomplished by sharing the Delegator's calendar with the Delegatee, with priviledges for both reading and writing or reading only.

The second calendar application we have exploited is the Exchange Service. It allows access to calendars hosted on a Microsoft Exchange system. In the current implementation this service is assumed to be installed in the same trust domain of the Service Centric System being executed by the SCENE platform. Also the two share the same Authentication Authority thus enabling a Single Sign-On approach. Based on this, the same credentials the user provides to `XTRIP` can be used to recognize the user also on the calendar. In this case no delegation is needed since the Exchange Service trusts the SCENE platform and allows it to access to the data of users who, based on the information stored in the Authentication Authority, have authorized it.

During the execution of `XTRIP`, the Proxy, after an initial processing to find the binding to the appropriate calendar service, invokes the IdM adapter. This last one checks if there is a Security Policy stored in the registry, linked to the bound service. Then, the Idm Adapter calls the Delegation Framework (through the Credential Provider Service) in order to obtain a delegation to access the calendar information of the requesting user (the delegator). The Delegation Framework asks the user for delegation and then, if the user agrees and selects the appropriate card, it calls the plugin responsible for the management of the access control mechanism protecting the calendar service. The Google Calendar plugin exploits the Google API in order to share the calendar of the requesting user with the SCENE platform account on Google Calendar. After the sharing is set, the IdM Adapter calls the service adapting the SOAP message, as required by the access control policy.

## 7 Related Work

Identity management and especially authentication and authorization management need to be improved to tackle the challenges of cross-domain service composition. This paper focuses on authorization and assumes associated authentication mechanisms. Multiple initiative exist to unify authentication mechanisms, e.g. OpenID [6] and Project Concordia [7], and to offer unified user experience, e.g. CardSpace [8] and Higgins project [9]. However, this state of the art focuses on authorization and delegation of rights in distributed system and service composition.

Some research works deal with access control in service composition. For instance, the approaches proposed in [17, 18] enable dynamic authorization for SOA. Wimmer et al. [19] support our reasoning, that "when integrating autonomous sub-activities into workflows, security dependencies must be

[6] See `http://openid.net`
[7] See `http://projectconcordia.org`
[8] See `http://netfx3.com/content/WindowsCardspaceHome.aspx`
[9] See `http://www.eclipse.org/higgins`

considered" and illustrate this with an e-health scenario. Such approaches are however mainly suitable for single trust domain SOA and impose a specific authentication mechanism.

It is broadly accepted [15, 20, 21, 22] that application-specific access control mechanisms are not suitable to compose services from different trust domains. In multi trust domain cases, there is clearly a trend to move to standardized mechanisms. XACML (Extensible Access Control Markup Language) [9] and SecPAL (Security Policy Assertion Language) [15] are recent declarative policy languages to express access rights. Both policy languages enable delegation of rights in distributed system and can be covert by the Delegation Framework. However, those policy languages do not enable dealing with the composition of legacy systems using other access control and management mechanisms. OAuth (Open Authentication) [23] and Windows Live ID Delegated Authentication [24] propose delegation mechanisms for managing access rights in unified ways, but they focus on web sites. Finally, Yu [25] also proposes to separate access control from services in order to reuse access control and management infrastructure.

Even if there is a trend to aggregate and standardize access control and delegation mechanisms, authors of this paper are convinced that multiple solutions will remain and that an abstraction layer able to manage many of them is necessary. The only work directly related to the abstraction layer proposed in the Delegation Framework is from Lang et al. [26]. They abstract a policy decision point with an object-oriented interface, which supports abstract operations to give access. Our approach offers an improved flexibility by adding new resource cards and plug-ins without touching the application.

## 8 Conclusion

In this paper we have presented our approach to IdM that consists in the integration of a Delegation Framework within a platform, SCENE, that supports the design and execution of self-adaptable Service Centric Systems. Such integration enables flexible adaptation of composite services by abstracting them from the need to know the service requirements in terms of Security Policies and Resource Cards.

In Section 6 we have shown how in a composed service various IdM needs may arise. Our approach addresses all of those in which users want to keep their personal services under control, but, at the same time, want to allow composed services to access personal services in a controlled way. Moreover, the approach can coexist with others, for instance, based on the federation of trust domains.

Our Delegation Approach does not require specific design effort to developers of a composed service. The Delegation Framework is automatically invoked by the runtime support when needed and takes care of all needed actions, from the request to the user for obtaining a delegation, to the actual invocation of the required service.

As future work we plan to continue experimenting with various application cases to verify the generality of our approach. Moreover, we are studying the integration of other IdM solutions in our architecture to be able to address as many IdM cases as possible.

## Acknowledgments

We thank Fabio Alloni, Giovanni Biscaro, Ali Muhammad and Uli Pinsdorf who have contributed to this work.

Parts of the work were sponsored by the European Commission in course of IST FP6 project SeCSE and of the IST FP7 NoE S-Cube. The technical details and opinions expressed in this paper are the authors' point of view and not necessarily the one of Microsoft Corp., the project participants in SeCSE and S-Cube, or the European Commission.

## References

1. Tziviskou, C., Di Nitto, E.: Logic-based management of security in web services. In: IEEE SCC, pp. 228–235. IEEE Computer Society, Los Alamitos (2007)
2. SeCSE: SeCSE IST Project, http://secse.eng.it
3. Colombo, M., Di Nitto, E., Mauri, M.: Scene: A service composition execution environment supporting dynamic changes disciplined through rules. In: Dan, A., Lamersdorf, W. (eds.) ICSOC 2006. LNCS, vol. 4294, pp. 191–202. Springer, Heidelberg (2006)
4. Papazoglou, M.: The challenges of service evolution. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074. Springer, Heidelberg (2008) (keynote address)
5. CEFRIEL, EMIC, L.T.: A4.d14 state of art and impact analysis of identity management. Report, SeCSE project (May 2007), http://www.secse-project.eu/wp-content/uploads/2007/09/a4d14-state-of-the-art-and-impact-analysis-of-identity-management.pdf
6. CEFRIEL, EMIC, L.T.T.: A4.d16 design of the 3nd version of the secse delivery platform. Report, SeCSE project (September 2007), http://www.secse-project.eu/wp-content/uploads/a4d16-design-of-the-3nd-version-of-the-service-delivery-platform.zip
7. SeCSE Consortium: Design of the 3rd version of the SeCSE delivery platform (focused on IdM). Public report A4.D19, SeCSE Project (February 2008), http://secse.eng.it/wp-content/uploads/.
8. Nadalin, A., Goodner, M., Gudgin, M., Barbir, A., Granqvist, H.: OASIS WS-Trust 1.4. Specification Version 1.4, OASIS, Currently in draft status, refer to version 1.3 for latest approved version (February 2008)
9. Moses, T.: OASIS eXtensible Access Control Markup Language (XACML) Version 2.0. OASIS Standard oasis-access_control-xacml-2.0-core-spec-os, OASIS (February 2005)
10. Cavallaro, L., Di Nitto, E.: An approach to adapt service requests to actual service interfaces. In: SEAMS 2008: Proceedings of the 2008 international workshop on Software engineering for adaptive and self-managing systems, pp. 129–136. ACM, New York (2008)
11. Active Endpoints: The ActiveBPEL Community Edition Engine, http://www.activevos.com/community-open-source.php

12. JBoss: Drools, http://www.jboss.org/drools/
13. Di Penta, M., Esposito, R., Villani, M.L., Codato, R., Colombo, M., Di Nitto, E.: Ws binder: a framework to enable dynamic binding of composite web services. In: ICSE Workshop on Service-Oriented Software Engineering (IW-SOSE 2006) (2006)
14. Ellison, C., Frantz, B., Lampson, B., Rivest, R., Thomas, B., Ylonen, T.: Rfc 2693 – spki certificate theory (1999)
15. Becker, M.Y., Gordon, A.D., Fournet, C.: Secpal: Design and semantics of a decentralized authorization language. Technical Report MSR-TR-2006-120, Microsoft Research (September 2006)
16. Google: Google calendar, http://www.google.com/calendar
17. Robinson, P., Kerschbaum, F., Schaad, A.: From business process choreography to authorization policies. In: [27] pp. 297–309 ISBN 978-3-540-36796-3
18. Mukkamala, R., Atluri, V., Warner, J., Abbadasari, R.: A distributed coalition service registry for ad-hoc dynamic coalitions: A service-oriented approach. In: [27] ISBN 978-3-540-36796-3
19. Wimmer, M., Kemper, A., Rits, M., Lotz, V.: Consolidating the access control of composite applications and workflows. In: [27], pp. 44–59 ISBN 978-3-540-36796-3
20. She, W., Thuraisingham, B., Yen, I.L.: Delegation-based security model for web services. In: Proceedings of 10th IEEE High Assurance Systems Engineering Symposium (HASE 2007), pp. 82–91. IEEE Computer Society, Los Alamitos (2007)
21. López, G., Cánovas, O., Gómez-Skarmeta, A.F., Otenko, S., Chadwick, D.W.: A Heterogeneous Network Access Service Based on PERMIS and SAML. In: Chadwick, D., Zhao, G. (eds.) EuroPKI 2005. LNCS, vol. 3545, pp. 55–72. Springer, Heidelberg (2005)
22. Freudenthal, E., Pesin, T., Port, L., Keenan, E., Karamcheti, V.: dRBAC: Distributed role-based access control for dynamic coalition environments. In: Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS 2002), Washington, DC, USA, pp. 411–420. IEEE Computer Society, Los Alamitos (2002)
23. OAuth Core Workgroup: OAuth Core 1.0. Technical report (2007)
24. Anonymous: Understanding Windows Live delegated authentication. White paper, Microsoft Corporation (February 2008), http://msdn2.microsoft.com/en-us/library/cc287613.aspx
25. Yu, W.D.: An intelligent access control for web services based on service oriented architecture platform. In: Proceedings of the The Fourth IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, and the Second International Workshop on Collaborative Computing, Integration, and Assurance (SEUS-WCCIA 2006), pp. 190–198. IEEE Computer Society, Los Alamitos (2006)
26. Lang, B., Foster, I., Siebenlist, F., Ananthakrishnan, R., Freeman, T.: A multipolicy authorization framework for grid security. In: Fifth IEEE International Symposium on Network Computing and Applications, pp. 269–272. IEEE Press, Los Alamitos (2006)
27. Damiani, E., Liu, P. (eds.): Data and Applications Security 2006. LNCS, vol. 4127. Springer, Heidelberg (2006)

# A Dynamic Orchestration Model for Future Internet Applications

Giuseppe Avellino[2], Mike Boniface[1], Barbara Cantalupo[2], Justin Ferris[1], Nikolaos Matskanis[1], Bill Mitchell[1], and Mike Surridge[1]

[1] University of Southampton IT Innovation Centre
2 Venture Road, Chilworth Science Park, Southampton, SO16 7NP
{mjb,nm,bm,ms}@it-innovation.soton.ac.uk
[2] Elsag Datamat spa, Via Laurentina 760, 00143, Rome, Italy
{giuseppe.avellino,barbara.cantalupo}@elsagdatamat.com

**Abstract.** Society and business are demanding systems that can securely and cost-effectively exploit opportunities presented by an Internet of Services. To achieve this goal a system must dynamically adapt to its environment and consider multiple and shifting stakeholder concerns such as application functionality, policies and business processes. In this paper we describe a dynamic orchestration model called the Virtual Infrastructure Model (VIM) which allows consumers to develop service-oriented systems that adapt to the needs of different business actors. It is based on the idea that adaptive workflow and dynamic binding to services can facilitate abstraction of both business processes and requisite interactions with the underlying infrastructure. Key requirements for federated orchestration are addressed including runtime service binding, secure and accountable dynamic procurement, infrastructure adaption, and separation of stakeholder concerns. The VIM is a fundamental component of the Next Generation Grid Architecture developed in the context of the EU funded NextGRID project.

**Keywords:** SOA dynamics, orchestration, lifecycle, workflow, business processes.

## 1 Introduction

After more than 10 years research and development of service-oriented systems, an economically viable Internet of Services have yet to materialise. Current software engineering theories, service specification and composition approaches assume software lifecycle models that significantly restrict the potential of service-oriented systems and the ability of such systems to support meaningful and dynamic social and economic relationships between communities and business partners.

Service-oriented systems are formed through the “recruitment” of services, possibly provided by different organizations, which are then orchestrated to achieve a desired objective. They cannot be subjected to conventional design “in advance”, so developers are left to create parts of a system that must fit together in ways that cannot be anticipated until run-time. Future service-oriented systems will therefore need to be operated by business stakeholders rather than developed by engineers. To achieve this operational model new orchestration approaches are required that support

© Springer-Verlag Berlin Heidelberg 2008

multiple stakeholder interactions allowing them to manage the lifecycle of assets and interact with other stakeholders flexibly and dynamically whilst considering distributed policy and regulatory compliance.

In this paper we describe an orchestration framework to makes service-oriented systems adaptable to the needs of different business stakeholders. This framework has been termed the Virtual Infrastructure Model and has been designed and developed in the context of EU funded NextGRID project [1]. The architecture is based on the idea that adaptive workflows and dynamic binding to services can facilitate abstraction of both business processes and requisite interactions with underlying infrastructure separating functional system aspects from the business processes that govern service interactions. A prototype of the VIM has been implemented and reference scenarios in several application domains have been developed to validate effectiveness of the approach in real business contexts.

## 2 Dynamic Service-Oriented Systems

The ability to select and use services from a variety of independent sources and to integrate them into a system that delivers the functionality and performance desired is required for dynamic service-oriented systems. In any service-oriented architecture (SOA) the key functional components are services and systems. The fact that systems can be considered to be composed of services, which services themselves are provided by systems, is important. The recursive self-referential characteristic of SOAs is why they are so powerful. However, it also means that service-oriented systems can quickly become extremely complex concealing lower level structures from end-users.

The complexity is dramatically increased when systems are built from an Internet of Services that incorporates a multitude of federations between service consumers and providers. Business relationships are generally codified in contracts making all relevant details explicit such as defining what is to be provided at what service level, relevant business practices and standards to be used, as well as pricing and penalties for failing to meet the specified conditions. In an Internet of Services, these terms are expressed in Service Level Agreements (SLAs) that identify the business context for relationships between systems and services and determine many of the technical policies that govern the interactions.

Federation is established in service-oriented systems by introducing business processes that result in federation contexts (SLAs) that provide a link between *access to service* and the *management the service*. To achieve federation in a dynamic way each of these aspects need to specified separately allowing systems to be built independently of the business models for provision and procurement of services from which they are composed. By introducing a separation of concerns, multiple stakeholder objectives can be supported and used to govern systems as they are operated. In addition, service providers will host different infrastructures with different business policies, and this may be true even when they offer the same service functionality. When consumers need their system to achieve a functional goal services need to be selected at runtime from multiple, sometimes competing, service providers. It is the dynamic orchestration of business relationships that supports the delivery of system functionality in a secure, trustworthy and accountable way that will provide an essential enabler to an economically viable Internet of Services.

## 3 The Workflow Landscape

Workflow is a critical technology for the orchestration of the interactions between systems and services. Workflow is important because it can be considered as the programming language for service-oriented systems and therefore has the potential to support process flexibility by soft-coding system behavior. In a SOA context, workflow is used to express a composition of services and there are several competing standards, initiatives and many more proprietary solutions.

The most widely used specifications for describing procedural workflows within businesses are XPDL [2] and ebXML [3], and the most widely used workflow specification with reference to SOAs is WSBPEL [4]. The focus of BPEL, and most business-oriented workflow languages, is control flow. However, extensive research on workflow control patterns has shown that all languages have limitations in terms of what can be easily expressed [5]. This insufficient expressivity and lack of rigorous semantics significantly limits their ability to support adaptation mechanisms and dynamics. Van de Aalst provides an extensive pattern comparison of workflow languages and implementations. Whilst the post-hoc evaluation of existing workflow languages against workflow patterns with well defined semantics is useful, it does not address the problem of inherent lack of rigorous machine-interpretable semantics within each workflow language.

The semantic web service community, on the other hand, is producing rigorous models and logics for the semantic description of Web Services. Several European projects inc. SEKT, DIP, SUPER, ASG are working together through the European Semantic Systems Initiative and have collaborated to develop the Web Service Modelling Ontology (WSMO) [6] and Web Service Modelling Language (WSML) [7]. Meanwhile work done by academia and industry through SWSI has resulted in the Semantic Web Service Framework (SWSF) [8], which has both a language (SWSL) [9], and an ontology (SWSO) [10] (based on OWL-S) that includes a process model. These languages and models make workflows more amenable to machine reasoning, making it easier to create abstract representations of processes and runtime binding to the services that incorporate the both functionality and QoS.

As far as we can ascertain, very few of the current approaches are considering the need to consider dynamic stakeholder concerns in workflow orchestration and as far as we can establish none have attempted to design and implement a complete architectural model addressing all the issues described in section 3.

## 4 Virtualised Infrastructure Model

The vision of the VIM is to provide a run-time adaptable infrastructure that meets the key requirements for dynamic systems operating in an Internet of Services, in particular:

- **Run-time bindings:** system workflows need not specify a binding of every task to a specific service, so that the bindings can be chosen at run-time.
- **Selective enactment:** a single service may provide multiple functions, and it must be possible to choose which is bound to an abstract task, supported by the service.
- **Workflow substitution:** some abstract tasks may be bound at run-time to more detailed workflows that can be inserted into the enactment at run-time. A common

example is substitutions with template business operations such as account and billing workflows.

- **Workflow prioritization**: Critical processes, which are either expensive in resources or define the result or the performance of the workflow, must have high priority in the evaluation order.

A key feature of the VIM's approach is the abstraction of business processes so developers do not have to encode business processes explicitly in their systems. This allows systems to remain functional even if a service provider wants to use a different business model or process (e.g. pay-as-you-go instead of subscription-based access to services). The result is a workflow enactment model with a corresponding workflow enactment engine that provides a way to dynamically assemble system functionality using an abstract application workflow specification as a starting point, and introducing business processes at run-time as specified by the service providers and consumers involved in executing the application.

The capability is achieved by combining adaptive semantic workflow, semantic discovery and service selection heuristics with supporting business and security services that govern functional services. System logic can be captured with abstract "application workflows" that include the functional constraints of the system. Service providers can publish workflows to describe the interactions and preconditions necessary for a consumer to use their service. During workflow execution, abstract tasks are resolved to concrete implementations including business process steps through a process that includes discovery, selection and rewiring, before execution.

## 4.1 Workflow Enactment Model

The overall enactment model for the VIM is illustrated in Figure 1. At its core the VIM provides an ***Enactor*** that is based on "evaluate - apply" cycles, as used in functional

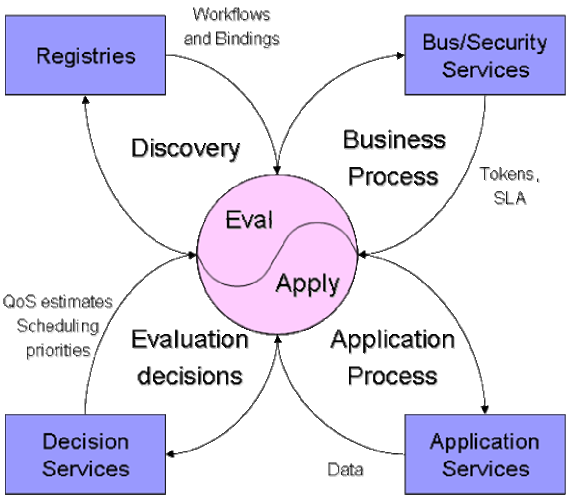


**Fig. 1.** VIM Enactment Model

programming. The aim of the evaluation is to replace abstract service descriptions with concrete services at runtime using components of the environment. Evaluation produces "concrete" processes that are either concrete ***Application Services*** or sub-workflows, which may contain abstract processes that also need to be evaluated. The apply phase that follows, executes the realized concrete processes. The evaluation algorithm includes four phases: prioritisation, candidate discovery, federation context acquisition, and candidate selection.

The evaluation order of a set of abstract processes in a workflow is determined by both enactor evaluation policies and prioritisation. Evaluation policies dictate whether to perform lazy or eager evaluation of conditional expressions, or when and how to fully evaluate nested composite workflows. Prioritization assigns priority weights to the workflow graph. Abstract processes with highest priority are evaluated first. Abstract processes that share the same priority level are evaluated together. Prioritisation helps the enactor to locate problems with availability of bindings for the abstract processes on critical parts of the workflow (e.g. missing SLAs). It also allows the enactor to to optimize execution by considering dependencies and data/control constraints.

Once prioritisation is completed candidate bindings for abstract processes are discovered from one or more ***Registry Services*** within the consumer's organisation.

In order to execute a candidate a consumer may need to acquire a federation context from supporting ***Security and Business Services.*** For example, if a SLA cannot be found, the enactor will use negotiation to establish a new SLA. SLA negotiations may also be required if service discovery fails to find any candidates. The negotiation of new SLAs can then allow access to more services, and when the SLA is agreed these services are added to the candidates list.

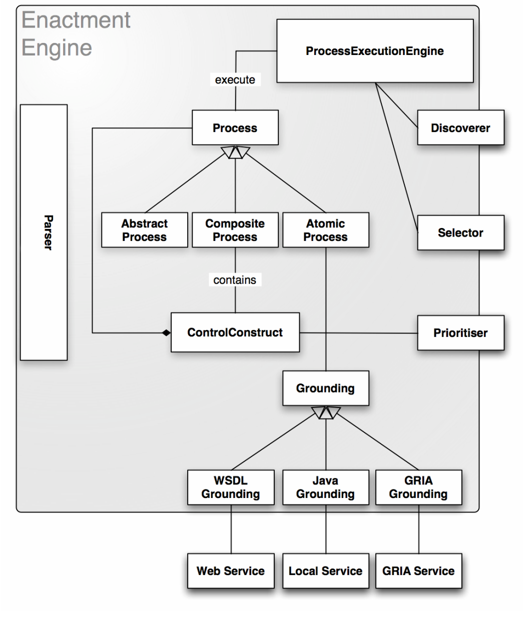


**Fig. 2.** Enactment Engine

Lastly the run-time binding of each abstract task to one of the candidates is determined using ***Decision services*** that apply selection heuristics and local organization policies. Selection operates across the whole workflow and may take account of co-location and other constraints. After selection of a candidate, replacement is made by rewiring the workflow and the ("Apply") phase is executes the task.

The workflow representation language that we adopted to represent workflows is OWL-WS. OWL-WS stands for "OWL for Workflows and Services" and is a workflow (and service) ontology fully based on OWL-S [11]. OWL-WS extends the OWL-S concept of Service to Abstract Process (an Atomic Process without implementation information), and uses the OWL-S concept of Composite Process for workflow modelling. In OWL-WS, Profile is available to any Process providing the ability to represent information at any level of the workflow composition. A more detailed description of the language is provided in [12]. A detailed model of the VIM can be seen in Figure 2.

## 4.2 Workflow Enactment Engine

The workflow enactment model has been implemented by integrating a range of service-oriented technologies. The ***Enactor*** is based on the Mindswap OWL-S API [13],which supports representation and enactment of OWL-S elements. Mindswap was extended to provide additional features to support OWL-WS extensions, and more complex and dynamic eval/apply execution semantics.

The evaluation order of an Abstract Process can be set both manually by the workflow author and automatically by the ***Prioritizer*** component. The Prioritizer uses QoS and historical information to assign evaluation priorities to those abstract processes that have not yet been prioritised. The ***Discoverer*** component looks for candidate bindings for an abstract process starting from its Profile. The Profile expresses constraints on the discovery process, effectively encapsulating a query that should be used to locate candidates. Discovery is performed by querying service registries that are located in the consumer's domain. This registry implementation is based on the Globus GT4 implementation of the WSRF-SG specification, and supports the XPath query language [14].

Federation context acquisition is implemented using the ***SLA Discovery*** and ***Broker*** components. The SLA Discovery component retrieves SLAs from a SLA registry. Many concrete services can only be executed under an agreement with the service provider, so an SLA reference is essential for the execution of these services. Negotiation of new SLAs with service providers is performed by means of the Broker component. The broker uses service provider registries to look for advertised services that fulfill consumer requirements. Once these have been decided the negotiation process takes place in order to establish a new SLA. The SLA that is produced is then registered to the SLA Registry followed by an update to the application service registry to register new service functionality that has been procured. The current Broker implementation consists of five components: the Matchmaker, the Negotiator, the Reconciler, the Template Retriever, and the Deal Closer and is described in detail in [15].

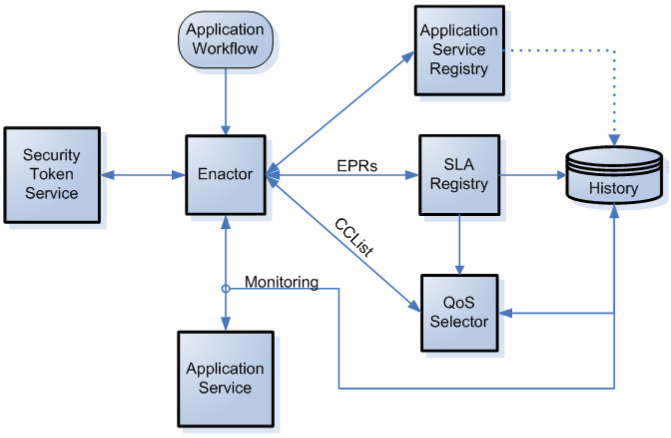


**Fig. 3.** VIM Components

The final selection of service bindings is performed by a ***Selector*** component. The selection process involves choosing a single candidate for each Abstract Process from the set of candidates found in the discovery phase. The selector implements an algorithm that takes into account criteria taken from an SLA, historical data and service parameters. User hints may also influence the selection. In order to determine the best from the available candidates for each step the selector takes into account services selected in other nodes of the workflow. The enactor gathers information of the service performance under a SLA, and can then be submitted to a Quality of Experience analysis service if the user so chooses. The QoE analysis can then also be used to produce criteria for the Selector component, based on previous experience of the candidate service providers [16].

Apart from the evaluation components the enactor uses Groundings to infrastructure implementations. These Groundings encapsulate the information required to construct and send appropriate messages to services and other executable components that are external to the Enactor. The WSDL and Java groundings enable Web Service and local service invocations. The GRIA grounding supports services hosted by the GRIA middleware [17], while the NextGRID grounding provides similar functionality but supports NextGRID specifications for SLAs and exchanged messages.

## 5 Real Context Experiments

The VIM architecture has been verified by architectural experiments and used in reference applications within the NextGRID project [18]. These applications include:

- Digital Media (DM): This application uses workflows that consume Rendering services for a for a television advertising company.
- Electronic Data Record (EDR). This application uses Grid services for a telecommunications company.

In the following paragraph we chose to analyse the Digital Media experiment. In the DM scenario, users want to run video rendering application workflows that have been written with the OWL-WS workflow-authoring tool by the application system experts. These workflows are abstract. The users have control over which workflow to use, over its input data and over the parameters for the execution through a web-portal. Parameters can specify preferences on price, availability, required time or other business factors. The application workflow in this scenario is shown in Figure 4:

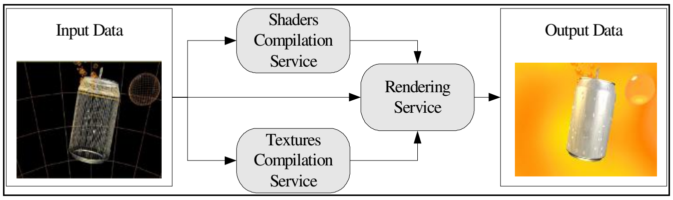


**Fig. 4.** 3D Video Rendering Scenario

Each of the abstract processes in this workflow is resolved by discovery and selection to a concrete application process with the SLA EPR information. The discovery performed in an order defined by the prioritiser implementation and the selection is taking into account the SLA terms, user preferences and Quality of Experience analysis results. The selection is done through out the workflow to take into account co-allocation issues according to the selector implementation.

This experiment demonstrated the ability of the system, through its user interface, to setup the environment of VIM infrastructure and enact abstract workflows of the video rendering application with different inputs and parameters. By changing the QoE parameters, users influenced selection and led to different concrete workflows that had different business models. In any case the abstract processes were evaluated by the VIM and concrete workflows with business management rules and policies were successfully enacted using NextGRID compliant application services.

## 6 Conclusions and Future Outlook

In this paper we presented an orchestration architecture for Future Internet applications based on a dynamic and adaptive workflow model. The architecture addresses the key requirements for service-oriented systems operating in an Internet of Services (runtime binding to services, secure and accountable dynamic procurement, infrastructure adaption, and separation of stakeholder concerns). A prototype workflow engine with related components has been developed and validated in significant business applications demonstrating how the lifecycle of system functionality and business processes governing underlying services can be separated.

The current implementation is limited to adapting consumer systems to service provider business processes by injecting these processes into application workflows at runtime. As we move towards an Internet of Services, consumers require systems that deal with the increased complexity and allow them to assess and mitigate threats in a more open world. To deal with these issues multiple business stakeholders (operations, finance, legal, quality, and marketing) will govern interactions and will work together to achieve an overall business objective. Effectively the atomic view of a consumer or service provider business process will be insufficient as multiple consumers will need to orchestrate their perspectives in goal-oriented event driven approach. This will require more fine-grained adaptive workflows to manage the lifecycle of different aspects of systems and services.

The creation and governance of applications of service-oriented infrastructures must become much easier for all stakeholders as the diversity and scale of assets dramatically increases, especially for applications that span multiple administrative domains. The VIM orchestration model introduces dynamics into service-oriented systems in a way that could not be previously achieved. Future work will continue to focus on orchestrating federations and will examine how the VIM model can be enhanced by applying functional programming and process algebra approaches to dynamic service composition and agent-based functionality in decision services.

## References

1. Next Generation GRIDs Expert Group Report 3, Future for European Grids: GRIDs and Service Oriented Knowledge Utilities (January 2006)
2. Workflow Management Coalition Workflow Standard, XML Process Definition Language (XPDL), Document Number WFMC-TC-1205 FINAL: Version 2.0, October 3 (2005)
3. OASIS standard v2.0.4, ebXML Business Process Specification Schema Technical Specification v2.0.4 (December 2006)
4. Alves, A., et al. (eds.): Web Services Business Process Execution Language Version 2.0, OASIS Committee Specification (January 2007)
5. Workflow Control-Flow Patterns A Revised View. Nick Russell, Arthur H.M. ter Hofstede (BPM Group, Queensland University of Technology) and Wil M.P. van der Aalst, Nataliya Mulyar (Department of Technology Management, Eindhoven University of Technology)
6. `http://www.wsmo.org/`
7. `http://www.wsmo.org/wsml/`
8. `http://www.w3.org/Submission/SWSF/`
9. `http://www.w3.org/Submission/SWSF-SWSL/`
10. `http://www.daml.org/services/swsf/1.0/swso/`
11. Martin, D. (ed.): OWL-S: Semantic Markup for Web Services, W3C Member submission (November 2004)
12. Beco, S., Cantalupo, B., Giammarino, L., Matskanis, N., Surridge, M.: OWL-WS: A Workflow Ontology for Dynamic Grid Service Composition. In: 1st Int. Conf. on e-Science and Grid Computing (2005)
13. See Mindswap OWL-S API project, `http://www.mindswap.org/2004/owl-s/api/`
14. Hasselmeyer, P.: On Service Discovery Process Types. In: Benatallah, B., Casati, F., Traverso, P. (eds.) ICSOC 2005. LNCS, vol. 3826, pp. 144–156. Springer, Heidelberg (2005)
15. Hasselmeyer, P., et al.: Towards Autonomous Brokered SLA Negotiation. In: Paul, Cunningham, M. (eds.) Exploiting the Knowledge Economy - Issues, Applications, Case Studies, vol. 3. IOS Press, Amsterdam (2006)
16. McKee, P., Taylor, S.J., Surridge, M., Lowe, R., Ragusa, C.: Strategies for the Service Marketplace. In: Veit, D.J., Altmann, J. (eds.) GECON 2007. LNCS, vol. 4685, pp. 58–70. Springer, Heidelberg (2007)
17. GRIA Middleware for Service Oriented Collaborations for Industry and Commerce, `http://www.gria.org/`
18. NextGRID application fliers, Digital Media application, `http://www.nextgrid.org/download/flyers/NextGRID%20Digital%20Media%20Flyer.pdf`, Electronic Data Records application, `http://www.nextgrid.org/download/flyers/NextGRID%20Digital%20Media%20Flyer.pdf`

# Defining the Behaviour of BPEL$^{\text{light}}$ Interaction Activities Using Message Exchange Patterns

Jörg Nitzsche, Benjamin Höhensteiger, Frank Leymann, Mirko Sonntag, and Markus Tost

Institute of Architecture of Application Systems
University of Stuttgart
Universitaetsstrasse 38, 70569 Stuttgart, Germany
{nitzsche,leymann}@iaas.uni-stuttgart.de
{hoehenbn,sonntamo,tostms}@studi.informatik.uni-stuttgart.de
http://www.iaas.uni-stuttgart.de

**Abstract.** BPEL$^{\text{light}}$ is an extension of BPEL that allows defining executable business processes independant of WSDL port types and operations. However, it adopts BPELs principle of having either non-blocking activities that only send or receive a single message or blocking activities, that are restricted to at most two messages, i.e. they implement a send-receive or receive-send behaviour. In recent work BPEL$^{\text{light}}$ has been used to define arbitrary complex message exchange patterns. In this paper we use message exchange patterns defined in BPEL$^{\text{light}}$ to describe the behaviour of interaction activities in a generic manner. This is beneficial as complex behaviour like a "request-for-bid" only have to be modelled once on an abstract level and can then be reused by simply referencing the corresponding message exchange pattern and filling in parameters whenever needed. This makes process modelling more convenient as the modelling primitives are not restricted to a request-response behaviour but are lifted to a business oriented level.

## 1 Introduction

Workflow technology [1,2] has become a very successful area in industry and research as it enables programming on a higher, i.e. business process oriented level [3] by separating business process logic and implementation of business functions. Process orientation has been discussed for many years but with the emergence of Web Services [4,5] (WS) which is the most popular implementation of a service oriented architecture [6,7] (SOA) workflow technology got established to a great extent. A workflow in general comprises 3 dimensions: process logic ('what' is to be done), organization ('who' does it) and infrastructure ('which' tools are used). In the Business Process Execution Language [8] (BPEL), which is part of the WS standard stack, the 'what' and 'which' dimensions are strongly coupled. Activities which are an aspect of the process logic ('what') directly refer to operations ('which') defined using the Web Service Description Language [9]. This ties BPEL to WSDL for referring to activity implementations and inhibits the reuse of processes or parts thereof in different contexts with different partners.

© Springer-Verlag Berlin Heidelberg 2008

BPEL$^{\text{light}}$ [10] gets over these deficiencies by defining a new interaction model using BPEL's extensibility. Thus, the interface of BPEL$^{\text{light}}$ processes and the interfaces of business functions used to implement activities of BPEL$^{\text{light}}$ processes can be described via any interface definition languages, i.e. it can be used even in non-WS environments (*WSDL-less BPEL*). The interaction model of BPEL$^{\text{light}}$ adopts BPELs principle of having either non-blocking activities that only send or receive a single message or blocking activities, that are restricted to at most two messages, i.e. they implement a send-receive or receive-send behaviour which is similar to that of WSDL 1.1 operations.

In recent work [11,12] BPEL$^{\text{light}}$ was used to define (WSDL 2.0 [13]) Message Exchange Patterns (MEPs). In this paper we use BPEL$^{\text{light}}$ MEPs to define the behaviour of BPEL$^{\text{light}}$ interaction activities in a generic manner. This enables the recursive definition of BPEL$^{\text{light}}$ MEPs and eases process modelling significantly. Instead of modelling for instance a request for bid behaviour using single message interaction activities every time it is required, an MEP can be modelled once which then can be reused to specify the behaviour of a complex interaction activity.

The remainder of the paper is structured as follows. Section 2 introduces BPEL and its extension BPEL$^{\text{light}}$. The application of BPEL$^{\text{light}}$ in the field of modelling MEPs is elaborated in section 3. In section 4 MEPs defined in BPEL$^{\text{light}}$ are used to describe BPEL$^{\text{light}}$ interaction activities and it is shown how this improves modelling processes and MEPs themselves. Section 5 concludes the paper and gives directions for future work.

## 2 BPEL and BPEL$^{\text{light}}$

BPEL has been approved as an OASIS[1] standard in 2007. It is the de facto standard for specifying business processes in a WS world and employs WSDL 1.1 to describe activity implementations. It enables both, the composition of WSs [4] and rendering the composition itself as WSs. Thus, BPEL provides a recursive aggregation model for WSs. The composition of WSs can be specified as a flow between operations of WS. According to the different operation types it provides several so called *interaction activities*. The control flow between activities can be structured either block-based by nesting *structured activities* like `<sequence>` (for sequential control flow), `<flow>` (for parallel control flow) and `<if>` (for conditional branches in the control flow) activities, or graph-based by defining `<links>` (i.e. directed edges) between activities in a `<flow>` activity; both styles can be used intermixed. BPEL does not support explicit data flow. Instead, data is stored in shared variables that are referenced and accessed by interaction activities and manipulation activities (e.g. `<assign>` activity).

As BPEL uses WSDL to define the operations a partner has to provide it lacks flexibility and reusability. To enable modelling flexible and reusable processes BPEL$^{\text{light}}$ [10] defines a new WSDL-less interaction model which completely decouples the process logic and interface definitions. The extensions

[1] `http://www.oasis-open.org/`

BPEL$^{light}$ defines in a seperate namespace[2] are (i) *interaction activities*, (ii) a `<bl:conversation>` that is referenced by a group of interaction activities to define a bilateral message exchange that is concerned with a certain business goal and (iii) a `<bl:partner>` that is used to define that several conversations have to take place with one and the same partner. All extensions do not reference WSDL elements, i.e. they are WSDL-less. The interaction activities include a basic `<bl:interactionActivity>` (see listing 1.1) that implements the behaviour of `<receive>` (receiving a message), `<reply>` (sending a message) and `<invoke>` (sending and then receiving a message) activities as well as first receiving and then sending a message, a `<bl:pick>` that implements a deferred choice and `<bl:eventHandler>`s. The behaviour of the `<bl:interactionActivity>` is defined via the attribute 'mode'.

```
<bpel:extensionActivity>
   <bl:interactionActivity name="NCName"
                           inputVariable="NCName"?
                           outputVariable="NCName"?
                           mode="in-out|out-in"?
                           conversation="NCName"
                           createInstance="yes|no"?
                           standard-attributes>
      standard-elements
   </bl:interactionActivity>
</bpel:extensionActivity>
```

**Listing 1.1.** BPEL$^{light}$'s `<interactionActivity>`

## 3 Formalizing MEPs Using BPEL$^{light}$

In contrast to WSDL 1.1, where a fixed set of operation types is defined, WSDL 2.0, which became a W3C[3] recommendation in 2007, defines operation types generically using so called message exchange patterns (MEPs). However, the expressivity of the formalism provided to define these MEPs is not sufficient because (i) it only allows defining a sequence of messages without any conditions, (ii) it does not distinguish partner node types and instances and (iii) it is too imprecise (it is not defined how the receipient of an optional message can find out whether the message will arrive or not). Therefore in [11] and [12] BPEL$^{light}$ was applied to modelling MEPs. In contrast to process models, MEPs do not define data types and are generically defined as they are aimed to be reusable [14]. This requires from BPEL$^{light}$ the possibility to define a flow between abstract messages that are received and sent without defining the concrete data type of these messages. Thus, an abstract BPEL$^{light}$ profile for MEPs was created [11].

[2] `xmlns:bl=http://iaas.uni-stuttgart.de/BPELlight/`

[3] `http://www.w3.org`

```
<bpel:process
   xmlns:bpel="http://.../wsbpel/2.0/process/abstract"
   xmlns:bl="http://.../BPELlight/"
   suppressJoinFailure="yes"
   abstractProcessProfile="http://.../bpel-light/abstract/mep/2008/"
   targetNamespace="http://../mep-in-bpel"
   name="request-wih-referral">
   <bl:conversations>
      <bl:conversation name="request-with-referral"/>
   </bl:conversations>
   <bl:partners>
      <bl:partner name="contacted-provider"/>
      <bl:partner name="responding-provider"/>
   </bl:partners>
   <bpel:flow>
      <bpel:links>
         <bpel:link name="L1"/>
      </bpel:links>
      <bl:interactionActivity name="Out" inputVariable="##opaque"
         partner="contacted-provider"
         conversation="request-with-referral">
         <bpel:sources>
            <bpel:source linkName="L1"/>
         </bpel:sources>
      </bl:interactionActivity>
      <bl:pick>
         <bpel:targets>
            <bpel:target linkName="L1"/>
         </bpel:targets>
         <bl:onMessage name="In" outputVariable="##opaque"
            partner="responding-provider"
            conversation="request-with-referral">
            <bpel:empty/>
         </bl:onMessage>
         <bl:onMessage name="InFault" faultName="##opaque"
            outputVariable="##opaque"
            partner="responding-provider"
            conversation="request-with-referral">
            <bpel:empty/>
         </bl:onMessage>
      </bl:pick>
   </bpel:flow>
</bpel:process>
```

**Listing 1.2.** “Request-with-Referral” MEP in abstract BPEL$^{\text{light}}$

This profile is associated with a namespace URI[4] and allows the use of all constructs allowed in the common base. The prefix associated with the MEP profile namespace URI is `mep`. Additionally it restricts the common base in the following manner[5]:

- Omission shortcuts (i.e. omitted elements) MUST NOT be used in the MEP profile with one exception: Timing definitions, i.e. `<for>`, `<until>`, and `<repeatEvery>`, MAY be omitted in `<onAlarm>` and `<wait>` elements. In this case, deadlines and durations MUST be defined by a newly introduced timing expression element which is necessary to enable expressing timing constraints without deciding whether they are defined via durations, deadlines or repetitions.
- Explicit opaque tokens, i.e. opaque activity, opaque attributes, opaque expression, and opaque from-spec, MUST NOT occur in MEP models except for variable references, types and time constraints. These opaque tokens denote the points of variability which have to be substituted later in order to come to a concrete meaningful form of this MEP process model.
- In order to define generic MEPs, data types MUST NOT be directly referenced by variables in MEP models. Instead, an opaque placeholder can be embedded which is to be replaced later. If message passing within the MEP process is not essential, `inputVariable` or `outputVariable` respectively can be marked opaque, the type is then automatically derived from the referenced messages. This way variable declarations can be omitted.
- Faults MUST be explicitly specified using BPEL$^{\text{light}}$'s `faultName` attribute in receiving activities (i.e. `bl:interactionActivity`, `bl:onMessage` within a `bl:pick`).

Using BPEL$^{\text{light}}$'s abstract profile for MEPs, an MEP can be defined in the following manner: For each MEP a seperate `<process>` definition is used. Within this definition an arbitrary flow between BPEL$^{\text{light}}$'s interaction activities (`<bl:interactionActivity>`, `<bl:pick>` and `<bl:eventHandler>`) is defined using BPELs control flow primitives. Note that for modelling MEPs only single message `<bl:interactionActivity>`s are used. Additionally a single conversation is defined to group these activities to an MEP.

Since MEPs are in general not restricted to be bilateral [14], the application of the `<bl:partner>` element in BPEL$^{\text{light}}$ has been modified. Instead of grouping whole conversations to indicate that they have to take place with one and the same partner, a `<bl:partner>` element is used to define for each message from which partner it is received or to which partner it is sent.

Listing 1.2 [12] presents a BPEL$^{\text{light}}$ description for the MEP *request-with-referral*. The activities or elements named "Out", "In" and "InFault" in combination with the corrsponding opaque variables represent the abstract messages.

[4] `http://iaas.uni-stuttgart.de/BPELlight/abstract/mep/2008/`

[5] The upper case keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [15].

They are grouped via the conversation named "request-with-referral". The partners "contacted-provider" and "responding-provider" indicate that both, the incoming message as well as the incoming fault, are received from the same partner which is different from the partner the first message was sent to. Additional sample MEPs can be found in [11] (in-only, robust-out-only, out-optional-in) and [12] (request-for-bid).

## 4 Describing Interaction Activities with MEPs

MEPs provide means to describe the flow of messages that are sent and received from the point of view of a service on an abstract level. They are used to describe the type of WSDL 2.0 operations. In BPEL, interaction activities correspond to the behaviour of WSDL 1.1 operations. A `<receive>` implements for instance a one-way operation, i.e. it only receives a single message, and an `<invoke>` behaves like a notification or solicit-response operation as it first sends a message and later may receive a message.

When first designed, BPEL$^{light}$ also adapted this behaviour for its interaction model, in particular for the `<bl:interactionActivity>`. However, as a task in a process – like an operation of a service – is supposed to provide support on a business oriented level by performing business functions like for instance a request-for-bid, it is reasonable to leverage reusable, abstract MEPs for defining the behaviour of interaction activities generically.

Therefore the syntax of the `<bl:interactionActivity>` has been refined (see listing 1.3). The attribute "mode" with its fixed values "in-out" and "out-in" has been replaced with the much more expressive attribute "mep" that points to any

```
<bpel:extensionActivity>
  <bl:interactionActivity conversation="NCName"?
                          createInstance="yes|no"?
                          mep="anyURI"?
                          standard-attributes>
    standard-elements
      <bl:input .../>*
      <bl:output .../>*
      <bl:infault .../>*
      <bl:outfault .../>*
      <bl:timingExpression .../>*
      <bl:partner .../>+
      ...
  </bl:interactionActivity>
</bpel:extensionActivity>
```

**Listing 1.3.** Syntax of `<interactionActivity>`

kind of MEP definition, including MEPs defined in BPEL$^{light}$. As an MEP may need several inputs and may provide several outputs the corresponding "input" and "output" attributes have been changed to elements with infinite cardinality. In addition, an MEP may need to send predefined faults and may receive faults. Hence, corresponding "infault" and "outfault" elements were defined. Apart from messages and faults an MEP parameterizes partner definitions and timing expressions used in `<onAlarm>` statements in `<bl:pick>`s or `<bl:eventHandler>`s. As an MEP may define several partners and timing expressions, the interaction activity provides elements with infinite cardinality for referencing them.

## 4.1 Recursive Definitions of MEPs

In recent work [11,12] BPEL$^{light}$ has been used to define MEPs. Within the given MEP definitions only single message interaction activities were used. Defining the bebaviour of interaction activities using MEPs, however, enables defining MEPs recursively, i.e. a MEP can be composed of several MEPs.

Listing 1.4 presents the refined syntax of the elements of the interaction activity including all attributes required to define MEPs recursively. The attribute "messageLabel" of the input and output element is used to distinguish all `bl:input` and `bl:output` elements defined within an abstract process that models a MEP from each other, i.e. messageLabels of input and output have to be unique throughout an MEP definition. The "messageRef" attribute is used to map an element to a corresponding input or output, defined in the MEP referenced via the "mep" attribute of the `<bl:interactionActivity>`. The same principle of identifying and referencing elements is applied to timing expressions. Once defined for a pick or eventHandler, the timing expressions are referenced from a `<bl:interactionActivity>`, referencing the MEP they are defined in, using the "timingExpressionRef" attribute. Within the `<bl:interactionActivity>` the element referencing the timing expression is in turn identified via the "timingExpressionLabel" attribute.

This enables the identification of all abstract elements throughout the recursive definitions, that are defined in the scope of a `<bl:interactionActivity>` and that need to be instantiated when used in a concrete process. Additionally,

```
    <bl:input messageLabel="NCName"?
              messageRef="QName"?/>*
    <bl:output messageLabel="NCName"?
               messageRef="QName"?/>*
    <bl:timingExpression timingExpressionLabel="NCName"?
                         timingExpressionRef="QName"?/>*
```

**Listing 1.4.** Syntax of `<interactionActivity>` elements for defining MEPs recursively

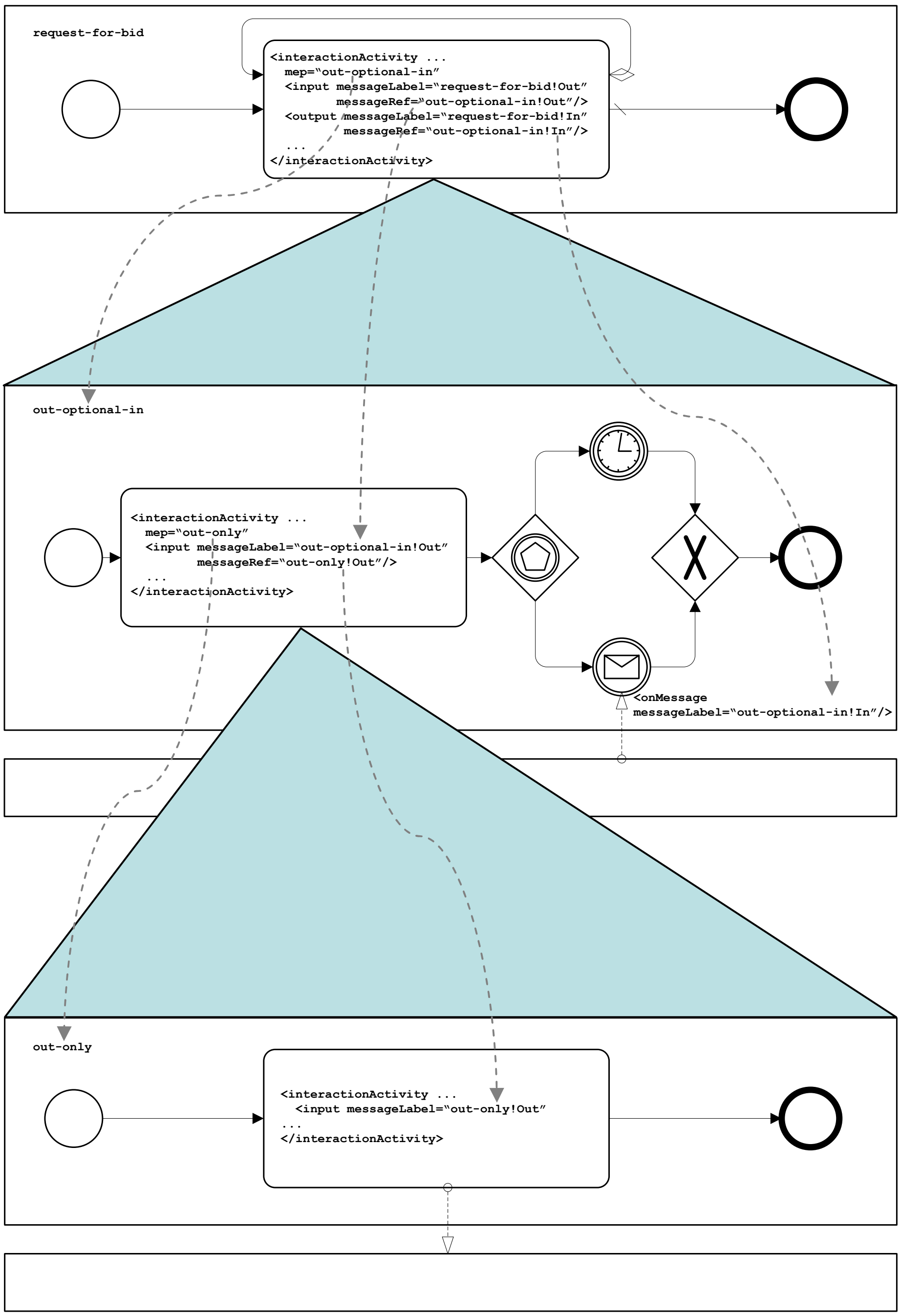

request-for-bid
<interactionActivity ...
mep="out-optional-in"
<input messageLabel="request-for-bid!Out"
messageRef="out-optional-in!Out"/>
<output messageLabel="request-for-bid!In"
messageRef="out-optional-in!In"/>
...
</interactionActivity>
out-optional-in
<interactionActivity ...
mep="out-only"
<input messageLabel="out-optional-in!Out"
messageRef="out-only!Out"/>
...
</interactionActivity>
<onMessage
messageLabel="out-optional-in!In"/>
out-only
<interactionActivity ...
<input messageLabel="out-only!Out"
...
</interactionActivity>

**Fig. 1.** Recursive Definition of "request-for-bid"

abstract defintions that are not defined in the scope of a single activity of a MEP but in the scope of the whole MEP need to be identified. This category comprises partner definitions as well as expression definitions needed to define the control flow by means of transition conditions for `<link>`s or conditions in `<if>` activities.

Figure 1 presents a scenario that illustrates the recursive definition of MEPs: the MEP “request-for-bid” is defined using a BPEL `<forEach>` performing several “out-optional-in” MEPs in parallel. Sending a message in this “out-optional-in” pattern is in turn defined using an “out-only” pattern. The figure uses the Business Process Modelling Notation (BPMN) [16] to visualize the control flow defined in BPEL$^{light}$. BPMN tasks represent `<bl:interactionActivity>`s. In addition, BPEL$^{light}$ code snippets are included in the BPMN diagram to show how the relationship of the different MEPs is defined.

The “out-only” pattern is defined by a single `<bl:interactionActivity>`. Within the activity one input with the messageLabel “out-only!Out” is defined. The MEP “out-optional-in” is defined by a `<bl:interactionActivity>` followed by a deferred choice pattern representing a `<bl:pick>` that consists of an event based gateway, an incoming message event representing a `<bl:onMessage>` event, a timer event representing an `<onAlarm>` event and an exclusive gateway. For the sake of simplicity, faults are not considered in this example. The `<bl:interactionActivity>` is specified by referencing the pattern “out-only”. Thus, it defines one input with the messageLabel “out-optional-in!Out” and maps to the abstract message definition in the “out-only” pattern via the messageRef attribute that points to the messageLabel “out-only!Out”. The message received via the `<bl:onMessage>` event is identified with the messageLabel “out-optional-in!In”. The “request-for-bid” pattern is defined using a loop including one task representing a `<bl:interactionActivity>`. This `<bl:interactionActivity>` references the “out-optional-in” pattern and defines one input and one output. The input is defined via the messageLabel “request-for-bid!Out” and maps to the abstract message definition in the “out-optional-in” pattern via the messageRef attribute that points to the messageLabel “out-optional-in!Out”. The output is defined accordingly.

Note, that a `<bl:interactionActivity>` can – instead of referencing the pattern “out-only” – also omit the “mep” attribute and define only one input. These two respresentations are semantically equivalent. The same applies to the pattern “in-only”.

## 4.2 Use in Concrete Process

When using MEPs to define the behaviour of `<bl:interactionActivity>`s in an executable process, concrete values have to be assigned to the abstract definitions. For defining the message types of inputs and outputs an additional variables attribute is sufficient (see Listing 1.5). Timing expressions can be configured according to the mechanism described in [11]. The same principle can also be applied to assigning concrete values to abstract definitions that are defined globally in the MEP like expressions and partners.

```
<bl:input variable="NCName"
          messageRef="QName"?/>*
<bl:output variable="NCName"
           messageRef="QName"?/>*
```

**Listing 1.5.** Assigning variable values to inputs and outputs of the `<interactionActivity>`

In case the MEP that is used to define the `<bl:interactionActivity>` has been defined recursively, it is not only possible to step-wise zoom into the MEP definition during modelling but it is also possible to suck in layer per layer into the main process. Once a layer is included in the main process, it is also possible to include additional control dependencies as long as they do not change the overall behaviour of the predefined MEP. In case a layer is drawn into the process, the conversation that defines a MEP becomes a conversation in the main process. As the relationship between the conversation the `<bl:interactionActivity>` was involved in and the conversation that defines the MEP used to specify the behaviour of this activity needs to be presereved, the conversation has been extended to enable pointing to a parent conversation. This way, BPEL$^{\text{light}}$ provides a broad spectrum of granularities of process models: from coarse grained business processes where a flow between `<bl:interactionActivity>`s with complex behaviour is specified to fine grained processes where the flow between single messages is visible.

## 5 Conclusion and Outlook

BPEL$^{\text{light}}$ is an extension of BPEL that allows defining executable business processes independent of WSDL port types and operations. In its first version it adopted the principle of having either non-blocking activities that only send or receive a single message or blocking activities, that are restricted to at most two messages, i.e. they implement a send-receive or receive-send behaviour.

In this paper reusable MEPs were applied to BPEL$^{\text{light}}$ to enable describing the behaviour of interaction activities generically. Therefore BPEL$^{\text{light}}$ and in particular the interactionActivity was extended to enable both, the recursive definition of MEPs using BPEL$^{\text{light}}$'s abstract profile for MEPs as well as using existing, reusable MEP definitions to describe the behaviour of *executable* processes. This eases process modelling as complex behaviour like a "request-for-bid" only has to be modelled once on an abstract level and can then be reused by simply referencing the corresponding MEP and filling in parameters whenever needed. The extensions shown make process modelling more convenient as the modelling primitives are not restricted to a request-response behaviour but are lifted to a business oriented level. This enables covering a broad spectrum of granularities of process models: from coarse grained business processes where a flow between `<bl:interactionActivity>`s with complex behaviour is specified to fine grained processes where the flow between single messages is visible.

To benefit from the expressivity provided by the new BPEL$^{light}$ interaction model tool support is required. To enable modelling MEPs as well as executable processes the eclipse BPEL designer[6], which is a GEF[7] based Eclipse plug-in that provides means to graphically model BPEL processes, is currently being extended to implement BPEL$^{light}$ extensions including the abstract profile for MEPs. In addition to the modelling tool, an execution environment is needed that is able to execute BPEL$^{light}$ processes. Therefore the Apache ODE engine[8] is being extended.

## Acknowledgements

The work published in this article was partially funded by the SUPER project[9] under the EU 6th Framework Programme Information Society Technologies Objective (contract no. FP6-026850).

## References

1. Leymann, F., Roller, D.: Production workflow. Prentice Hall, Englewood Cliffs (2000)
2. van der Aalst, W., van Hee, K.: Workflow management. MIT Press, Cambridge (2002)
3. Leymann, F., Roller, D.: Workflow-based applications. IBM Systems Journal 36(1), 102–123 (1997)
4. Weerawarana, S., Curbera, F., Leymann, F., Storey, T., Ferguson, D.: Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging and More. Prentice Hall PTR, Upper Saddle River (2005)
5. Alonso, G., Casati, F., Kuno, H., Machiraju, V.: Web Services: Concepts, Architectures and Applications. Springer, Heidelberg (2004)
6. Burbeck, S.: The Tao of e-Business Services. IBM Corporation (2000)
7. Krafzig, D., Banke, K., Slama, D.: Enterprise SOA: Service-Oriented Architecture Best Practices (The Coad Series). Prentice Hall PTR, Upper Saddle River (2004)
8. Alves, A., Arkin, A., Askary, S., Barreto, C., Bloch, B., Curbera, F., Ford, M., Goland, Y., Guizar, A., Kartha, N., Liu, C.K., Khalaf, R., König, D., Marin, M., Mehta, V., Thatte, S., van der Rijn, D., Yendluri, P., Yiu, A.: Web Services Business Process Execution Language Version 2.0. Committee specification, OASIS Web Services Business Process Execution Language (WSBPEL) TC (January 2007)
9. Christensen, E., Curbera, F., Meredith, G., Weerawarana, S.: Web Services Description Language (WSDL) 1.1. W3C Note (2001)
10. Nitzsche, J., van Lessen, T., Karastoyanova, D., Leymann, F.: BPEL$^{light}$. In: 5th International Conference on Business Process Management (BPM), Brisbane, Australia (September 2007)

[6] http://www.eclipse.org/bpel/
[7] http://www.eclipse.org/gef/
[8] http://ode.apache.org/
[9] http://www.ip-super.org/

11. van Lessen, T., Nitzsche, J., Leymann, F.: Formalising Message Exchange Patterns using BPEL$^{light}$. In: 5th International Conference on Services Computing (SCC), Honululu, Hawaii, USA (July 2008)
12. Nitzsche, J., van Lessen, T., Leymann, F.: Extending BPEL$^{light}$ for Expressing Multi-Partner Message Exchange Patterns. In: 12th IEEE International EDOC Conference (EDOC 2008), Munich, Germany (September 2008)
13. Chinnici, R., Moreau, J.J., Ryman, A., Weerawarana, S.: Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. W3C Recommendation (2007)
14. Nitzsche, J., van Lessen, T., Leymann, F.: WSDL 2.0 Message Exchange Patterns: Limitations and Opportunities. In: 3rd International Conference on Internet and Web Applications and Services (ICIW), Athens, Greece (June 2008)
15. Bradner, S.O.: Key Words for Use in RFCs to Indicate Requirement Levels. Internet RFC 2119 (March 1997)
16. White, S.: Business Process Modeling Notation (BPMN) Version 1.0. Object Management Group/Business Process Management Initiative, BPMN.org. (2004)

# Managing Technical Processes Using Smart Workflows

Matthias Wieland[1,⋆], Daniela Nicklas[2], and Frank Leymann[1]

[1] Institute of Architecture of Application Systems, Universität Stuttgart
wielanms,leymann@iaas.uni-stuttgart.de

[2] Computer Science Department, Carl von Ossietzky Universität Oldenburg
dnicklas@acm.org

**Abstract.** Technical processes that are crossing the boundary to the physical world can be found in many application domains, like logistics or in Smart Factory environments. We show how these processes can be realized by so-called Smart Workflows. To integrate external information sources like context provisioning services, we introduce the Integration Process architecture pattern. This pattern generally solves the problem of integrating different complex systems that provide functional similar services with non-fitting interfaces into workflows. The pattern allows that workflows use simple domain specific interfaces that are the same for any of these systems and by that allow the exchange of underlying systems without changing the workflows. This is accomplished by reducing the interface complexity of the systems via a hierarchical Web Service stack that reaches from the lowest technical granularity needed by IT experts to the domain specific granularity needed by the domain experts. Furthermore the paper presents a concrete realization of the pattern for integrating different context provisioning systems into workflows.

## 1 Introduction

Technical processes that are crossing the boundary to the physical world, like production processes or maintenance processes in a smart factory, are not well supported by workflow technology yet. Our vision is compared to the current state of the art to automatically execute and control such technical processes. This leads to the same amount of flexibility enterprises gained by introducing workflow management systems [1]. Furthermore, technical processes are enabled to easily interact with the back office, bridging the gap between business and production. Modeling a technical process becomes similar to modeling of a business process. The difference is that for modeling technical processes various information about the physical world like current state of all involved real world entities is needed, e.g., machines, tools, and workers in a production environment. If this information—often referred to as context [2]—is captured automatically by sensors in a smart environment, technical processes can be executed pervasively and

⋆ This work was funded by the Collaborative Research Center *Nexus: Spatial World Models for Mobile Context-Aware Applications* (grant SFB 627).

© Springer-Verlag Berlin Heidelberg 2008

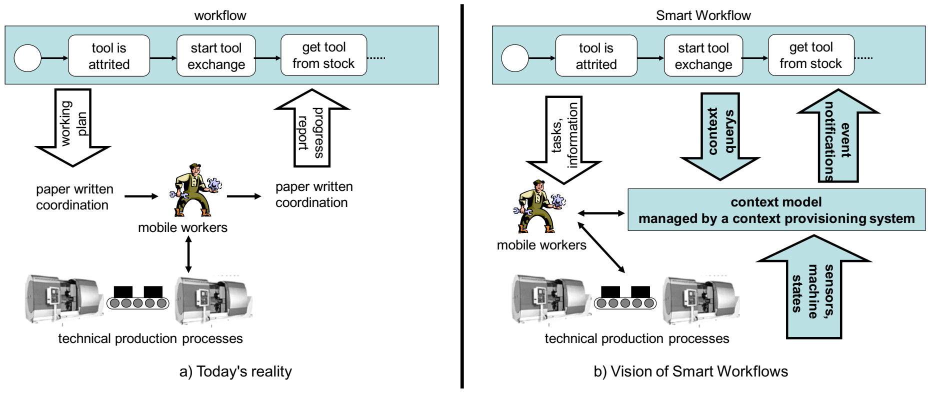


**Fig. 1.** Smart workflows: Incorporate context information into workflow technology

adapt to changes in their physical context. We call such context aware processes *Smart Workflows* (SW). To realize this vision we employ context data managed in a *context model* (see Figure 1). In Section 1.1 we illustrate a sample scenario in a Smart Factory [3].

While designing a service oriented system, a main challenge is to choose an adequate granularity of the services. It could be either generic and rather technical (like complex SQL queries) or domain-specific and rather semantic for a given application domain (like functions for retrieving machine tool information for a set of parameters). This choice cannot be made in general. Also there are no standards available how services can be re-used, e.g., in a service hierarchy where services are derived from each other (comparable to a class hierarchy in a programming language).

For coping with both problems we developed the integration process architectural pattern. Integration processes (IPs) are processes that are derived from each other, i.e., their interfaces get more restricted and thereby more concrete and domain specific. With that concept it is possible to build up a hierarchical Web Service stack that reaches from the lowest technical granularity needed by IT experts to the domain specific granularity needed by the domain experts.

In the remainder of this section we describe a concrete scenario and derive challenges and requirements. After a discussion of the related work in Section 2 we present the main contribution of this paper, the concept of IPs, in Section 3. In Section 4 we show a concrete realization of that concept, the context integration processes (CIPs). Finally, we describe the prototype implemented for evaluation and proof of concept in Section 5 and conclude the paper in Section 6.

## 1.1 Example Scenario: Machine Maintenance Process

The following process is used as concrete example throughout the paper and represents the kind of processes we implement with Smart Workflows (SWs): A sensor in a machine measures the attrition of an installed tool, a drill. When the tool is attrited, a SW starts to arrange the replacement of that tool. First, the SW finds out whether a spare drill is in stock. If not, it starts a business process in the back office that orders new tools and registers a notification for when the spare part is available. Now that a spare part is in stock, the SW creates a human task: somebody should transport the spare part from the storage to the measuring device to prepare it for installation. The exact location of the spare part within the storage and the available measuring device are part of this task. Also, the SW monitors the execution of this task using location information. Soon, a mobile agent (a transport robot or a worker) that is near to the storage picks up this task. When the transport task is completed, the SW creates a preparation task which can be picked up by another agent that is capable of doing this work. Again, the completion of this task is monitored so that a transport task to the machine can be invoked. Arriving there, the old drill can be exchanged by the spare part. The last action triggers an event for the tool life-cycle SW of the old drill that now checks whether the old drill can be refitted or has to be recycled.

### 1.2 Challenges and Requirements

To realize scenarios as the one given, several challenges must be addressed:

*1. Smart workflow modeling:* A domain expert should be able to easily model SWs. For that, new concepts for workflows are needed, like context-based decisions, or context information requests. These concepts require the access of context data. Since the domain expert should not be burdened with unnecessary technical details (like the syntax of a general context query language), the context access for SWs needs a high number of simple, domain-specific functional interfaces. Here, we need a design that offers good maintainability of these domain-specific functions because they are often extended to new application needs. Finally, the SWs must be executed in an efficient manner. The easiest way would be to use existing workflow execution engines. Thus, existing standards like BPEL [4] should be used for modeling SWs whenever possible.

*2. Context provisioning:* We need an environment to capture, manage, and provide context information in an efficient way. This information is distributed over various systems within the factory. Also, it varies regarding update rate, selectivity, usage for selection, and required data quality. Hence, a single server solution is not feasible. In the Nexus project[3], we developed a model-based, extensible context provisioning platform that is able to cope with high amounts of distributed context data [5]. Since context modeling and management is expensive, the main goal is to provide reusable context information for different applications, like in a shared database. Thus, the interfaces of such a platform are rather generic: for context information requests, a flexible query language is provided; for event-based interaction, complex physical world events can be declared using an event definition language.

*3. System integration:* Since the context access functions used for SW modeling are often domain-specific they should be understandable by domain experts.

[3] Nexus project website: `http://www.nexus.uni-stuttgart.de/index.en.html`

Hence, there is a gap between generic provisioning of context (e.g., query languages) and the concepts needed in the SW (e.g., "spare tool available?"). This context access functions must be organized in an extensible modular architecture that allows the easy reuse of existing functions. Also, they should be realized using the same modeling techniques as for the SWs to allow domain experts to easily extend the available domain-specific functional interfaces.

## 2 Related Work

IBM's information integration for BPEL (II4BPEL) [6] allows simple and efficient access of relational database systems, using SQL, from within business processes. Other vendors provide comparable SQL support in BPEL [7]. These solutions could be used to integrate context information into SWs. However, only BPEL engines implementing this extension could be used. Our approach works with every engine conforming to the BPEL standard.

The PerCollab System [8] extended workflow technology to support adaptive collaboration between people in business processes. While this work can be used to enhance the collaboration of workers in a shop floor, it does not take the context of the tasks or activities of the process into account. In [9], a ubiquitous workflow service framework for the development of context aware services named uFlow is introduced, which defines an own workflow description language and workflow engine. Again, the usage of standard BPEL allows us to leverage the advantages of existing workflow engines, e.g., the process management or server stability.

There are many approaches specialized on the handling of context data like the Context Toolkit [10], the Location Stack [11], or the CML based approach by Henricksen et al. [12]. They provide simple software components to access sensor data, to refine the raw sensor data to some higher level and for the interpretation of situations based on the sensed data. In our approach we need a more powerful context model which offers us the ability to model and access a wide range of common objects in a unified manner. We also model our applications as workflows, thus we are rather focused at accessing software components defined as Web Services.

In previous work we proposed an extension to BPEL to cope with the special needs of context aware workflows, called Context4BPEL [13]. To execute Context4BPEL workflows an extended workflow engine is needed. The SW approach in this paper can be either build on top on Context4BPEL or using only standard BPEL. In contrast to the Context4BPEL approach [13], this paper presents a full end-to-end solution providing both domain-specific workflow functions needed by domain-experts and a generic execution architecture using standard BPEL workflow technology. Nevertheless the CIPs on the higher layers, especially the domain-specific CIPs are needed in both cases because it is not feasible to define a workflow modeling language that contains all domain specific extensions. In [14] we presented a short work in progress overview of the SW vision and the layered system architecture needed for realization.

## 3 Concept of the Integration Processes Pattern

The concept of Integration Processes (IPs) addresses the service granularity problems described in Section 1. On Figure 2 the generic design of the Integration Process architecture pattern is shown.

At the bottom of the architecture the *systems* that provide generic services (data and functionality) are shown (system A, system X). These systems are the artifacts of integration. The aim is to access them from the top most part of the architecture, the *service users* (here: a Smart Workflow (SW)). The SW represents the technical processes executed in the production environment in our scenario. For defining SWs, we chose the Business Process Execution Language (BPEL) [4]. This language can be used to orchestrate Web Services. The SWs use the functionality provided by the IPs analog to external Web Service invocation. In future work, the BPEL extension for subprocesses [15] will be used to invoke the IPs.

The *IPs* are the main part of the architecture. They are located between the systems and the users and are implemented similarly as BPEL workflows. Hence, they are accessible as Web Services. There are three different granularity levels for IPs: core, domain independent, and domain specific IPs. The lowest level processes are the *core IPs*. They are responsible for the integration of each service the systems provide. Each service has its own interfaces and one concrete core IP that wraps that interface.

On the highest level the *domain specific IPs* are located. The SWs on the application layer use them to access the services of the systems in a simplified manner. In general, the domain specific IPs receive fewer parameters than the other IPs. The aim of this design is that domain experts can easily model SWs using only the domain specific IPs. The interface parameters of these IPs derive from the application area and thus form the terminology of the domain experts. If functionality is missing, the domain expert could create a request for

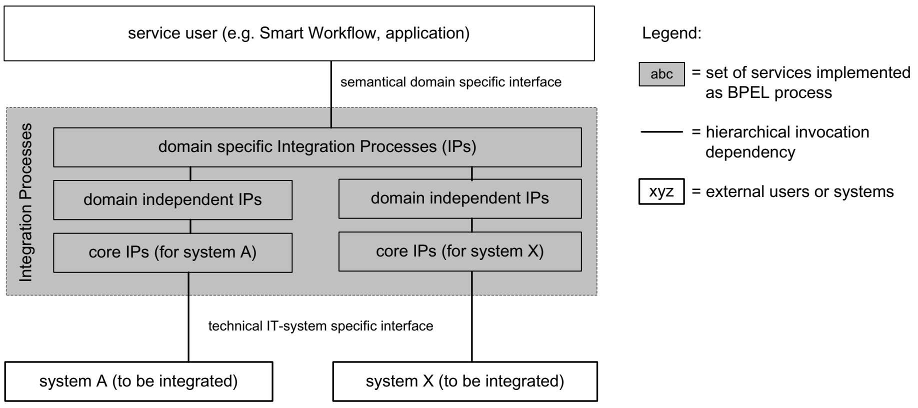

service user (e.g. Smart Workflow, application)
semantical domain specific interface
Integration Processes
domain specific Integration Processes (IPs)
domain independent IPs
domain independent IPs
core IPs (for system A)
core IPs (for system X)
technical IT-system specific interface
system A (to be integrated)
system X (to be integrated)
Legend:
abc = set of services implemented as BPEL process
= hierarchical invocation dependency
xyz = external users or systems

**Fig. 2.** Architectural pattern for the concept of integration processes

that interface and an information scientist could implement this functionality by deriving from and thus reusing existing domain independent or core IPs.

The most important level is located in between the other two layers. The *domain independent IPs* mediate between the core IPs with complex interfaces and the domain specific IPs with easy-to-use interfaces. The domain independent IPs can be reused in different application domains because they offer a general and not application specific functionality. The main functionality of the IPs on this layer is the transformation from the simple request/response message formats of the IPs on the higher levels to the more complex request/response message formats of the IPs on the lower level.

The reusability of services is further enhanced by the hierarchical structure of the pattern. This has the following advantages: if the interfaces of system A are changed, only the core IPs wrapping that interfaces have to be changed. The process does not notice the changes and even the IPs on the higher level do not have to be changed. In addition, if a new interface is requested by a domain expert it can be provided easily by attaching it on top of an appropriate existing IP.

The hierarchical structure should be part of the naming scheme of the processes, i.e., the processes on the core layer should have a name describing their functionality (e.g., FunctionA). The processes on the domain independent layer should add their specialization characteristics to the end of the name of the core IP they are using (e.g. FunctionAFiltered). In contrast, the processes on the domain specific layer should use names from the application domain and not technical functionality descriptions (e.g. QueryTool).

## 4 Realization of Context Integration Processes

To solve the challenges described in Section 1.2 we implemented following system based on the Integration Processes pattern. On system level we use the Nexus Context Provisioning System, which solves challenge 2 (context provisioning). On the user level we use BPEL or Context4BPEL [13] for modeling the technical processes in the Smart Factory, which addresses challenge 1 (Smart Workflow modeling). The integration of the Nexus system (challenge 3) or other context provisioning systems is realized by a set of implemented Context Integration Processes (CIPs) described in Section 4.2.

### 4.1 Nexus Context Provisioning System

To provide the highly dynamic context information for SW in an adequate manner, a mature context management is needed. We use the Nexus platform for that purpose. It is an open platform that supports the development of various context aware applications. It is based on a common context model, the so-called Augmented World Model (AWM). This context model is used to integrate and cache the highly dynamic context information from various sensor sources and to provide an abstraction for different context aware applications [5]. It models

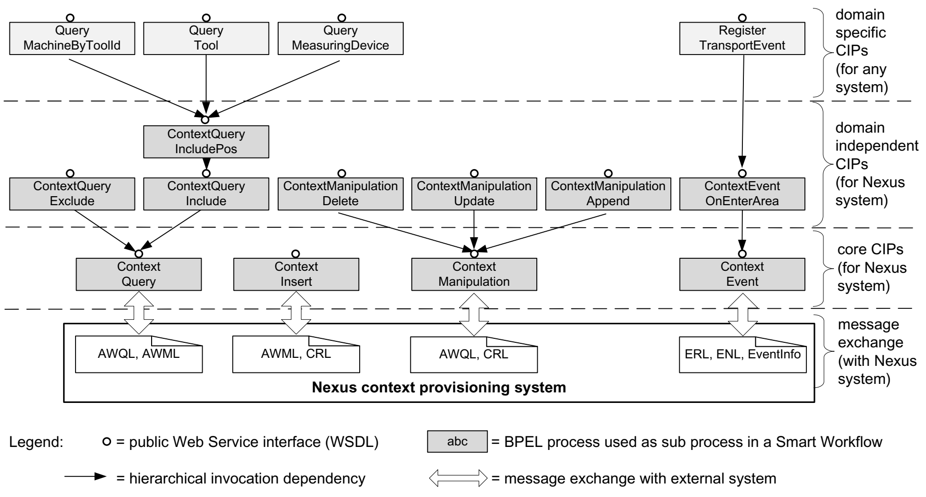

Query MachineByToolId
Query Tool
Query MeasuringDevice
Register TransportEvent
domain specific CIPs (for any system)
ContextQuery IncludePos
ContextQuery Exclude
ContextQuery Include
ContextManipulation Delete
ContextManipulation Update
ContextManipulation Append
ContextEvent OnEnterArea
domain independent CIPs (for Nexus system)
Context Query
Context Insert
Context Manipulation
Context Event
core CIPs (for Nexus system)
AWQL, AWML
AWML, CRL
AWQL, CRL
ERL, ENL, EventInfo
Nexus context provisioning system
message exchange (with Nexus system)
Legend:
= public Web Service interface (WSDL)
abc
= BPEL process used as sub process in a Smart Workflow
= hierarchical invocation dependency
= message exchange with external system

**Fig. 3.** Context Integration Processes for the Nexus context provisioning system

context data in different areas, like geographical data, dynamic sensor data, infrastructural context, or related information such as documents. Nexus consists of basic services and value added services to provide the context information. Nexus is a federated system and there exist many different kinds of implemented and ready to use context servers for diverse needs [5]. To exchange context information between applications and the Nexus platform, several data formats have been defined [16]:

- the Augmented World Modeling Language (AWML) for data modeling and serialization of context information;
- the Augmented World Query Language (AWQL) for querying and manipulating context information. The results for manipulations (success or error) are reported with the Change Report Language (CRL);
- the Event Registration Language (ERL) and the Event Notification Language (ENL) for context event observation;
- the map service can be used to generate topographic maps based on the objects (e.g. buildings) stored in the context servers. As exchange formats the map service offers the Map Predicate Language (MapPL) and the Map Modeling Language (MapML); and
- the navigation service is responsible for calculating travel routes and offers the Navigation Parameter Language (NPL) and the Navigation Result Language (NRL) for exchanging data.

## 4.2 Context Integration Processes

Figure 3 shows as a concrete implementation of the Integration Process pattern for the integration of the Nexus services. We call this concrete set of IPs Context

Integration Processes (CIP) hence it is used for integration of context data into Smart Workflows (SW).

As *core CIPs* we need the following four processes:

- *ContextQuery* is used for querying context data. It integrates the querying functionality of the Nexus federation component. ContextQuery integrates the AWQL format for the query declaration and the AWML format for the result presentation.
- *ContextInsert* is used for inserting new context data. It integrates the AWML format for describing data objects that have to be inserted to the context model and the CRL format for reporting the result of an insert operation.
- *ContextManipulation* is used for changes on existing context data. It integrates the AWQL format for the manipulation request and the CRL format for manipulation results.
- *ContextEvent* integrates the functionality of the event component from the context provisioning layer. It integrates the ERL format for the registration of events and the ENL format for event notifications (more detailed description in 4.2).

The CIPs on this level have the advantage of offering the full functionality of the integrated Nexus components by supporting the complex exchange formats of the Nexus components. However, for client processes (e.g., SWs) the interfaces of these CIPs are too complex for an easy usage.

Hence, following *domain independent CIPs* are defined that simplify these interfaces:

- *ContextQueryExclude* allows the blanking out of a set of attributes to minimize the size of the result set. It uses the core CIP ContextQuery for the query execution.
- *ContextQueryInclude* allows the selection of a set of attributes. Only this set of attributes is included in the objects of the result set. That can be used to downsize the messages that have to be transferred. It uses the core CIP ContextQuery for the query execution.
- *ContextQueryIncludePos* is used by QueryMeasuringDevice, QueryTool and QueryMachineByToolId. It returns just the location attribute of an object. It gets a restriction parameter that holds the condition for the objects that should be searched for. It uses the CIP ContextQueryInclude for further processing of the query.
- *ContextManipulationDelete, ContextManipulationUpdate, ContextManipulationAppend* are specializations of the core CIP ContextManipulation. They allow the deletion of complete objects in the context model, the update of already existing objects and the extension of existing objects with new attributes.
- *ContextEventOnEnterArea* is used by RegisterTransportEvent. It requires the ID of the mobile object, the observation period of the event, the geographic area that should be observed, a threshold probability for the event firing, and further configuration parameters (more detailed description in 4.2).

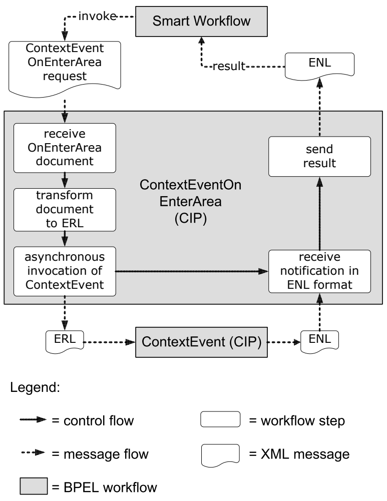

invoke
Smart Workflow
ContextEvent OnEnterArea request
result
ENL
receive OnEnterArea document
transform document to ERL
asynchronous invocation of ContextEvent
ContextEventOn EnterArea (CIP)
send result
receive notification in ENL format
ERL
ContextEvent (CIP)
ENL
Legend:
= control flow
= message flow
= BPEL workflow
= workflow step
= XML message

Fig. 4. ContextEventOnEnterArea

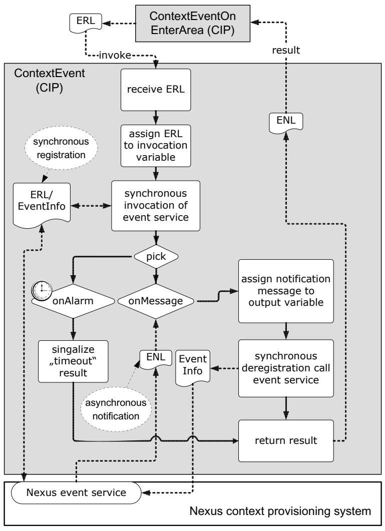

ERL
ContextEventOn EnterArea (CIP)
invoke
result
ContextEvent (CIP)
receive ERL
assign ERL to invocation variable
synchronous registration
ERL/ EventInfo
synchronous invocation of event service
ENL
pick
onAlarm
onMessage
assign notification message to output variable
singalize „timeout" result
ENL
Event Info
synchronous deregistration call event service
asynchronous notification
return result
Nexus event service
Nexus context provisioning system

Fig. 5. ContextEvent

For the Smart Factory application domain we modeled following *domain-specific CIPs* (see Figure 3):

- *QueryMachineByToolId* queries the machine where the tool has to be exchanged. This CIP takes the ID of the old tool that should be exchanged by the worker, and searches for the machine with this tool installed. The return value is the location of the machine.
- *QueryTool* queries a tool as spare part for a machine. This CIP takes a tool type as parameter and returns the exact geographic location of an available and not worn out tool in the factory.
- *QueryMeasuringDevice* queries a measuring device, where a tool can be calibrated by a factory worker. This CIP accepts only a location area parameter, e.g., the factory area, because it is intended to search measuring devices in a given area. It returns the location of the measuring device.
- *RegisterTransportEvent* observes the transportation process of a tool. It gets three parameters: an arbitrary mobile object, a geographic area as transport destination and the observation duration of the event.

**Detailed Realization of Two Example CIPs.** To illustrate the concrete realization, we show the detailed structure of the core CIP ContextEvent and the domain-independent CIP ContextEventOnEnterArea building up hierarchically on that core CIP. In Figure 4, we see the realization of the ContextEventOn-EnterArea CIP. It calls the ContextEvent CIP as a subprocess to register its specialized event (an object enters a specified area). Therefore it offers a simpler interface to clients by defining its own request format. The new request format holds, compared to the ERL format, only fewer parameters necessary for that special event type. Thus, a transformation step is needed before the invocation of the ContextEvent CIP to adapt the more complex ERL format. The received result from the ContextEvent CIP will be forwarded to the client. The client has thus unmodified access to the complete result data described in ENL format.

In Figure 5, the realization of the ContextEvent CIP is illustrated. A client process (e.g., the previous ContextEventOnEnterArea CIP) sends an event registration request in ERL format to the ContextEvent integration process. ContextEvent itself takes the message and sends it synchronously to the event service located on the Nexus platform. The request is send synchronously with an immediate response that confirms the registration of the event. However, the notification of the event occurrence is performed asynchronously. The ContextEvent CIP stores the current process state and waits for the event notification by suspending the process activity. When a notification arrives, the process wakes up and continues processing by forwarding the result to the client process. Alternatively, a timer activates the process if the event was not observed in a given time, thus allowing the client process to take appropriate action in this case. This CIP is used to integrate the event service of the Nexus platform by integrating the complex documents described in ERL and ENL.

This example showed the hierarchical approach of enhancing and reducing the complexity of the requests and responses and showed how the CIPs can be implemented as BPEL workflows. All other CIPs are implemented and wired the same way.

## 5 Implemented Prototypes

To evaluate the concept of SWs, we implemented two prototypes in the Smart Factory environment [3]. This example factory contains a storage area, a measuring device for drills, and some machines that use different drills for producing personalized plastic coins. Furthermore several sensors are available for context observation. The drills are equipped with RFID tags for identification. Tools are transported in an intelligent transport box. This box is tracked by an UbiSense[4] indoor positioning system. Different possible locations of the tools (i.e., the transport box, the storage, or the machine) are equipped with a RFID reader. Thus, the position of a specific tool is available anytime using indirect localization. Also, the usage time of drills within machines is measured. This allows the context management system to calculate the attrition of the tools. The whole factory layout—i.e., the positions of the machines and workstations—is managed by a Nexus context model. Also, the transport cart, which is used to transport the tool boxes, is tracked by the UbiSense system. Transport carts are moved only by workers. Hence, the system can infer the positions of workers without tracking them directly. This improves the acceptability of the system due to the enhanced privacy for workers.

Within this setup we implemented following two SWs: First, the machine maintenance process described in Section 1, and secondly, a process for handling

[4] UbiSense Real-time Location System: `http://www.ubisense.de/`

individual customer orders. For the execution of the modeled BPEL processes we used the Oracle BPEL process manager[5]. The provided Dashboard gives a good overview of all deployed, in-flight, and recently completed BPEL processes. Furthermore the Oracle BPEL process manager provides a Human Task Management Service with web interface which is used for the interaction with the workers. A worker can list and access all his ongoing tasks via the human task manager. Thus the workflows used in the Smart Factory can be controlled both directly by human interaction and pervasively by observing changes and events in the real world (e.g., movement of the transport cart). From our experience with these first prototypes we derived the great need for an easy maintainable and hierarchically structured service layer. So we invented the integration process architecture pattern and implemented the CIPs as BPEL workflows therewith. The most important benefit was that domain experts could now better understand and use the resulting workflows because of the mainly domain specific interfaces called by the SWs.

## 6 Conclusion

In this paper we introduced the notion of Smart Workflows (SWs), which cross the boundaries to the physical world. Many application areas like logistics or the upcoming domain of Smart Factories could benefit from workflow technology if these are enhanced by context information to SWs. To realize SWs, we need context-based features at the process modeling level, an efficient provisioning of context information, and a maintainable integration layer for providing the context information at the right semantical level.

Our prototype leverages the usage of standards like WSDL and BPEL, for both the realization of SWs and the provisioning of various context services by Context Integration Processes (CIPs), which are used to bind an off-the-shelf workflow engine (Oracle) with an existing context provisioning platform (Nexus). This binding is realized by implementing the Integration Processes pattern described in this paper. By splitting up different semantical layers (from general context access to highly application-dependent services) into a hierarchy of different CIPs, we achieved a high maintainability and reusability of the services. This approach dramatically facilitated the development of SWs, which is an important pre-condition for the adoption of that technology by industry. Like in business process engineering, SWs should be developed by domain experts using modeling techniques rather than be programmed by computer scientists. The usage of BPEL as a well known workflow modeling language speeds up the modeling of new SWs compared to using an application specific workflow language, or even against programming a context aware application supporting the process (for example in Java). It enables domain experts to model their SWs themselves. These resulting workflow models can be used very good as a basis for discussion between domain experts and the computer scientists.

[5] Oracle BPEL Process Manager `http://www.oracle.com/technology/bpel/`

## References

1. Leymann, F., Roller, D.: Production Workflow: Concepts and Techniques. Prentice Hall PTR, Englewood Cliffs (1999)
2. Dey, A.K.: Understanding and Using Context. Personal and Ubiquitous Computing 5(1) (2001)
3. Westkaemper, E., et al.: Smart Factory - Bridging the gap between digital planning and reality. In: Proc. of the 38th CIRP Intl. Seminar on Manufacturing Systems (2005)
4. OASIS: Web Services Business Process Execution Language Version 2.0 (2007), http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html
5. Großmann, M., et al.: Efficiently Managing Context Information for Large-Scale Scenarios. In: Proc. of the Third IEEE Intl. Conf. on Pervasive Computing and Communications (2005)
6. IBM: Information Integration for BPEL on WebSphere Process Server. (2005), http://www.alphaworks.ibm.com/tech/ii4bpel
7. Vrhovnik, M., et al.: An Overview of SQL Support in Workflow Products. In: Proc. of the 24th International Conference on Data Engineering (2008)
8. Chakraborty, D., Lei, H.: Pervasive Enablement of Business Processes. In: Proc. of the Second IEEE Intl. Conf. on Pervasive Computing and Communications (2004)
9. Han, J., Cho, Y., Kim, E., Choi, J.-Y.: A Ubiquitous Workflow Service Framework. In: Gavrilova, M.L., Gervasi, O., Kumar, V., Tan, C.J.K., Taniar, D., Laganá, A., Mun, Y., Choo, H. (eds.) ICCSA 2006. LNCS, vol. 3983, pp. 30–39. Springer, Heidelberg (2006)
10. Salber, D., Dey, A.K., Abowd, G.D.: The Context Toolkit: Aiding the Development of Context-Enabled Applications. In: CHI 1999: Proc. of the SIGCHI Conf. on Human factors in computing systems. ACM Press, New York (1999)
11. Hightower, J., Brumitt, B., Borriello, G.: The Location Stack: A Layered Model for Location in Ubiquitous Computing. In: Proc. of the Fourth IEEE Workshop on Mobile Computing Systems and Applications (2002)
12. Henricksen, K., Indulska, J.: A Software Engineering Framework for Context-Aware Pervasive Computing. In: Proc. of the Second IEEE Intl. Conf. on Pervasive Computing and Communications (2004)
13. Wieland, M., et al.: Towards Context-Aware Workflows. In: Pernici, B., Gulla, J.A. (eds.) CAiSE 2007 Proc. of the Workshops and Doctoral Consortium, vol. 2. Tapir Acasemic Press (2007)
14. Wieland, M., Kaczmarczyk, P., Nicklas, D.: Context Integration for Smart Workflows. In: Proc. of the Sixth IEEE Intl. Conf. on Pervasive Computing and Communications (2008) (work in progress paper)
15. Kloppmann, M., et al.: WS-BPEL 2.0 Extensions for Sub-Processes. Whitepaper, IBM, SAP AG (2005)
16. Bauer, M., et al.: Information Management and Exchange in the Nexus Platform. Technischer Bericht Informatik 2004/04, Universität Stuttgart, Universität Stuttgart, Institut für Parallele und Verteilte Systeme, Verteilte Systeme (2004)

# Model Driven QoS Analyses of Composed Web Services

Danilo Ardagna, Carlo Ghezzi, and Raffaela Mirandola

Politecnico di Milano, Dipartimento di Elettronica e Informazione,
Piazza Leonardo da Vinci 32, 20133 Milano, Italy
{ardagna,ghezzi,mirandola}@elet.polimi.it

**Abstract.** The problem of composing services to deliver integrated business solutions has been widely studied in the last years. Besides addressing functional requirements, services compositions should also provide agreed service levels. Our goal is to support model-based analysis of service compositions, with a focus on the assessment of non-functional quality attributes, namely performance and reliability. We propose a model-driven approach, which automatically selects the set of available services, transforms a design model of service composition into an analysis model, which then feeds a probabilistic model checker for quality prediction. To bring this approach to fruition, we developed a prototype tool and we show the results which can be achieved with a simple example.

**Keywords:** Web services composition, QoS, Business Process Optimization, Probabilistic Model. Checking.

## 1 Introduction

Service-Oriented Architectures (SOAs) provide a new paradigm for the creation of business applications. This paradigm enforces decentralized developments and distributed systems compositions: new added-value services may be created by composing independently developed services.

Web services (WSs) are an increasingly important and practical instance of SOAs and are supported by standards and specific technologies. Typically, services can be composed in an *orchestrated* manner by using a process language, like the *BPEL* [1].

We argue that SOAs can benefit from the Model Driven Development (MDD) [4] paradigm. In essence, this means that models are built to support software engineers in reasoning at the software architecture level. As a satisfactory solution is built at the model level, transformation steps (possibly automated) derive the final, platform-specific implementation. In the case of SOAs, model-level reasoning should support the early QoS assessment of a service composition. The composition may be assessed at design time, before a concrete binding from the workflow to the externally invoked services is established. Indeed, applications can be specified as abstract processes with unbounded invocations to external services. Concrete services can then be selected and bound at run time according to some optimizing strategy [27]. QoS assessment hence has to be performed on the abstract process. It is requested, however, that a specification of the external services in terms of their functional and non-functional attributes is available.

© Springer-Verlag Berlin Heidelberg 2008

The use of models extends beyond the initial design of an application. Models may be used to support both the initial derivation of an implementation and then an evolution of the software architecture. They can also be useful to devise suitable reconfiguration strategies for the dynamic contexts where the application will be deployed. Once the application is running, model-based reasoning may be used to predict the impact of different reconfigurations in a changing context, driving the reconfiguration process.

In this paper we propose an overall framework for the automatic service selection and QoS analysis of composed Web services. In particular, the service selection is formalized as a non-linear optimization problem which allows to determine an optimal solution for the execution of multiple composed services. The QoS analysis is built on stochastic models and architectural reasoning is performed through probabilistic model checking [6,7,22].

The paper is organized as follows. In Sections 2 and 3 we describe the proposed architectural evaluation framework and the composed process model. Section 4 introduces a reference example, while the two steps of service selection and analysis of the service composition are described in Sections 5 and 6 respectively. Section 7 briefly summarizes the related work and Section 8 concludes the paper.

## 2 The Architectural Evaluation Framework

In the system under study, service providers assure QoS guarantees and publish their profiles in a WS registry. It is then possible to select automatically the set of services to be invoked at run time according to varying workload conditions and end user QoS preferences. Our architectural framework, illustrated in Figure 1, is based on two main functional components: (i) a service optimizator based on SNOPT [24] that, starting from composed service descriptions, determines an optimal service selection satisfying a set of global QoS constraints; and (ii) a composed quality analyzer [15], which automatically derives stochastic models that can be solved by PRISM [22] to provide insights about the overall application quality.

In our approach, the software architect describes a set of applications he intends to realize and their functional and non-functional requirements as annotated abstract BPEL processes. Abstract processes are composed by *abstract services* which act as *place holders* of Web service components invoked at run time. In this way, the "best" set of services can be selected at run-time by solving an optimization problem. Web

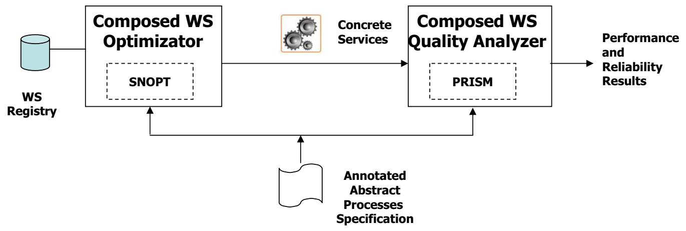

Composed WS Optimizator
SNOPT
WS Registry
Concrete Services
Composed WS Quality Analyzer
PRISM
Performance and Reliability Results
Annotated Abstract Processes Specification

**Fig. 1.** General Framework

services are then invoked by implementing a *dynamic/late binding* mechanism. To perform quantitative analyses, composed services specifications are also annotated in order to provide statistics on processes executions. Each composed process is then transformed in a Directed Acyclic Graph (DAG). Without loss of generality, we assume that BPEL processes have single starting and ending points, and the loops are peeled or unfolded before the analysis is computed as in [3,10,27]. Hence the DAG has a source node and a sink node. An execution of the composite service consists of the invocation of the services on a path from the source to the sink.

Several quality criteria can be associated with Web services execution. Furthermore, if the same Web service is accessible from the same provider, but with different quality characteristics, then multiple copies of the same Web service are stored in the registry, each copy being characterized by its quality profile. In the following, Web services will be indexed by $j$ and denoted by $ws_j$.

In this paper, we focus on a subset of quality dimensions, which have been the basis for QoS consideration also in other approaches [13,20,27].

– *Service Reliability* denoted by $r_j$, a real number between 0 and 1 that represents the reliability of the service invocation $ws_j$;
– *Service Execution Time* denoted by $e_j$, represents the expected execution time of the service invocation $ws_j$;
– *Service Cost*, denoted by $c_j$, which indicates the cost associated with the invocation of service $ws_j$.

This quality model can be extended in order to include other quality dimensions. Furthermore, each abstract service process specification is associated with the *Service Invocations Attempts* which represents the number of failed invocations necessary to declare a service to be faulty at run-time.

The composed service optimizator determines through SNOPT (a state of the art non linear solver [24]) the set of concrete Web services which will be invoked at run time. Then, the Composed Quality Analyzer module supports the evaluation of reliability and performance metrics by using the PRISM stochastic model checker.

In the next section we introduce the composed service model and the notation adopted in the framework.

## 3 Composed Service Model

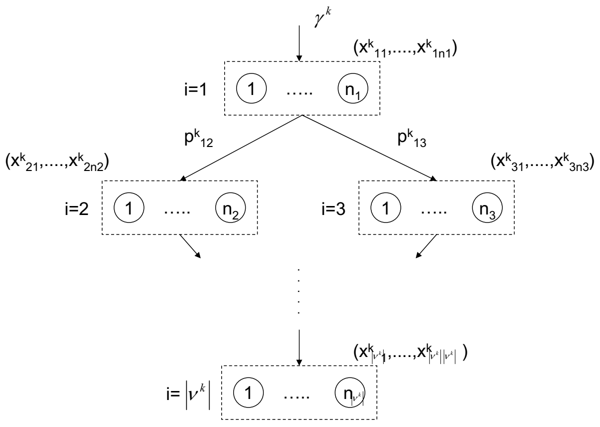
As discussed above, the quality profiles are stored in an extended UDDI registry. Furthermore, as in [19], we assume that the service providers (SPs) store the maximum service rate $\mu_j$, i.e., the maximum incoming workload which can be accepted by the service provider. As in grid environments [3], we assume that each SP pre-allocates some resources to the system in order to provide QoS guarantees. In the following we will model each service $ws_j$ as a M/M/1 queue [9].

The model of composed services adopted in this paper is driven by [11]. We denote by $K$ the set of QoS classes, by $\gamma^k$ class-$k$ requests incoming workload ($k \in K$), and by $\boldsymbol{\gamma} = (\gamma^1, ..., \gamma^{|K|})$ the overall user requests arrival rate to the system.

For each class-$k$ request, the Composed WS Optimizator has to assign a concrete service $ws_j$ for each abstract service $i$ in the DAG, under given global QoS constraints (i.e., constraints over the *overall composed service execution*).

Let be $\mathcal{V}^k$ the set of indexes of abstract services specified in class-$k$ process. We denote by $J_i^k$ the set of all concrete services $ws_j$, $j \in J_i^k$, that implement the abstract service $i \in \mathcal{V}^k$, and let be $\mathcal{J} = \bigcup\limits_{k \in K} \bigcup\limits_{i \in \mathcal{V}^k} J_i^k$.

In Figure 2, each macro-node depicted as a rectangular box represents an atomic *abstract service* $i \in \mathcal{V}^k$ in the DAG. The directed edge from the macro-node $r$ to the macro-node $s$ represents a sequencing constraint; that is, it indicates that service $r$ must complete before service $s$ may begin.



$\gamma^k$
$(x^k_{11},\ldots,x^k_{1n1})$
i=1
1 ..... $n_1$
$p^k_{12}$
$p^k_{13}$
$(x^k_{21},\ldots,x^k_{2n2})$
$(x^k_{31},\ldots,x^k_{3n3})$
i=2
1 ..... $n_2$
i=3
1 ..... $n_3$
$(x^k_{|\mathcal{V}^k|1},\ldots,x^k_{|\mathcal{V}^k||\mathcal{V}^k|})$
i= $|\mathcal{V}^k|$
1 ..... $n_{|\mathcal{V}^k|}$

**Fig. 2.** Example of DAG for class-$k$ process

Multiple edges exiting from a macro-node $r$ are weighted by a probability, which provides statistical information about the next abstract service required by a client of the composite service. The Composed WS Optimizator can use its observation of the execution patterns generated by the clients to estimate these probabilities. In this probabilistic model of the workflow execution pattern we do not include the parallel execution of services. We are currently working toward inclusion of parallelism in our model. In the following we denote with $p_{rs}^k$ the probability expressing the frequency with which service $s$ is executed after completion of service $r$ in process $k$. For each macro-node $r$, $\sum_{s \in succ(r)} p_{rs}^k = 1$. If only one edge exits node $r$, the probability is equal to 1 and we omit its value in the graph. Different service classes are associated with a different process schema and probabilities.

Let $\lambda_i^k$ be the rate of class-$k$ requests that arrive at the abstract service $i \in \mathcal{V}^k$. Using well-known flow conservation arguments [9], we get the following set of linear equations for the request rates, that can be used to calculate $\lambda_i^k$:

$$\boldsymbol{\lambda}^k = \mathbf{P}^{k\mathrm{T}} \boldsymbol{\lambda}^k + \gamma^k \mathbf{e}_1 \quad \forall k \in K \tag{1}$$

where $\boldsymbol{\lambda}^k = (\lambda_1^k, ..., \lambda_{|\mathcal{V}^k|}^k)$ and $\mathbf{e}_1 = (1, 0, ..., 0)$ are column vectors and $\mathbf{P}^k = [p_{rs}^k]$ is the $|\mathcal{V}^k| \times |\mathcal{V}^k|$ routing probability matrix for class-$k$ requests. In the following we will denote by $\lambda_i^{k*}$ the flow of requests for the abstract service $i$ given by the solution of equation (1).

Each DAG macro-node contains the *concrete services* (shown in Figure 2 as circles inside the rectangular box representing the abstract service), that correspond to specific implementations of a given abstract service.

Finally, each request class is associated with:

- a set of normalized weights $\{w_e^k, w_c^k, w_r^k\}$, $w_e^k + w_c^k + w_r^k = 1$, indicating a relative priority among the set of quality dimensions for class $k$ end users;
- the maximum (minimum) values of QoS required for the composed service invocation, i.e., maximum execution time $e_{max}^k$, maximum cost $c_{max}^k$ and minimum reliability $r_{min}^k$;
- class $k$ weight $\Omega^k$, which denotes the class $k$ relative priority, and $\sum_{k \in K} \Omega^k = 1$.

## 4 Reference Example

In this section, we introduce a simple case study which in the following will be used to exemplify the use of the proposed framework.

We consider 2 processes including 7 abstract services which can be supported by 11 concrete Web services (see Figure 3). Requests for the first and second composite service are classified as *silver* and *gold* classes, respectively.

The first composed service includes a simple sequence while the second introduces a switch. Each abstract service can be supported by two different candidate services while, in both cases, the last service is supported only by the concrete service $ws_5$.

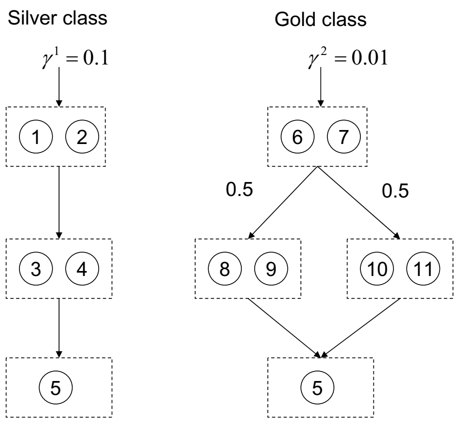


**Fig. 3.** A Simple Case Study

**Table 1.** Parameters of the example

| **Parameters** | **Values** |
|---|---|
| $(\gamma^1, \gamma^2)$ | (0.1, 0.01) |
| $(\mu_1, \mu_2, \mu_3, \mu_4, \mu_5, \mu_6,$ $\mu_7, \mu_8, \mu_9, \mu_{10}, \mu_{11})$ | (0.2, 0.4, 0.8, 0.5, 0.3, 0.3 0.4, 0.8, 0.9, 0.9, 0.3) |
| $(c_1, c_2, c_3, c_4, c_5, c_6,$ $c_7, c_8, c_9, c_{10}, c_{11})$ | (0.9, 1.3, 1.2, 0.3, 0.8, 1.1 0.5, 0.7, 1.2, 1.8, 2.6) |
| $(r_1, r_2, r_3, r_4, r_5, r_6,$ $r_7, r_8, r_9, r_{10}, r_{11})$ | (0.99, 0.999, 0.99, 0.999, 0.99, 0.999 0.999, 0.99, 0.999, 0.99, 0.999) |
| $(\Omega^1, \Omega^2)$ | (0.3, 0.7) |
| $(c^1_{max}, c^2_{max})$ | (3, 4) |

The routing probability matrices for the two processes are $\mathbf{P}^1 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$ and $\mathbf{P}^2 = \begin{bmatrix} 0 & .5 & .5 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$. Users in the gold class accept to pay a higher cost ($c^2_{max} = 4$) and are interested in the maximization of the process reliability, i.e., $w^2_r = 1$, while users in the silver class introduce stringent execution costs ($c^1_{max} = 3$) and are interested in the minimization of the execution time $w^1_e = 1$. Table 1 summarizes the system parameters.

## 5 Composed WS Optimizator

The goal of the Composed WS Optimizator is to select, for each QoS class-$k$, the concrete service $ws_j$, $j \in J^k_i$, that must be used to fulfill a request for the abstract service $i$ in order to maximize the QoS perceived by the end user and guaranteeing global constraints. The Composed WS Optimizator generally acts on behalf of many potential requestors, it is able to identify recurrent requests for typical service compositions, as well as usage patterns of these compositions. In our approach, service selection is performed probabilistically and constraints are guaranteed statistically. The decision variables of the problems are $x^k_{ij}$ which denote the probability that the concrete service $j \in J^k_i$ will be invoked by the class-$k$ request when the workflow reaches the stage indicated by the macro-node $i$.

Given a flow of requests $\lambda^{k*}_i$ for the abstract service $i$, the Composed WS Optimizator splits it among the corresponding concrete services $j \in J^k_i$ according to the $\mathbf{x} = [x^k_{ij}]$ probabilities. Hence, $x^k_{ij}\lambda^{k*}_i$ is the flow of requests for the concrete service $j$ generated by clients belonging to the QoS class $k$.

The global QoS that is experienced by class-$k$ users depends both on the total request flow addressed to each concrete service, and on the value of the service QoS attributes. The Composed WS Optimizator can affect this QoS by appropriately setting the $x^k_{ij}$ values, which are under its control. Given a matrix $\mathbf{x}$, we denote by $F^k(\mathbf{x})$ the corresponding global QoS for class-$k$. Under the M/M/1 assumption, the average execution time of each abstract service $i$ can be computed as:

$$exeTime_i = \frac{\lambda_i^{k*}}{\gamma^k} \sum_{j \in J_i^k} \frac{x_{ij}^k / \mu_j^k}{1 - \sum_{h \in K} \sum_{a \in \nu^h} \frac{x_{aj}^h \lambda_a^h}{\mu_j^h}} \tag{2}$$

In the following, we indicate with $E^k$, $C^k$, and $R^k$ respectively the average of the execution time, execution cost, and reliability for the composed service which can be evaluated as follows:

$$E^k(\mathbf{x}) = \sum_{i \in \mathcal{V}^k} exeTime_i \tag{3}$$

$$C^k(\mathbf{x}) = \sum_{i \in \mathcal{V}^k} \frac{\lambda_i^{k*}}{\gamma^k} \sum_{j \in J_i^k} x_{ij}^k c_j^k \tag{4}$$

$$R^k(\mathbf{x}) = \prod_{i \in \mathcal{V}^k} \frac{\lambda_i^{k*}}{\gamma^k} \sum_{j \in J_i^k} x_{ij}^k r_j^k \tag{5}$$

note that, $\lambda_i^{k*}/\gamma^k$ is the mean number of class-$k$ invocations to the $i$-th abstract service.

$C^k$ and $R^k$ are obtained as in [3,27] and are computed as the sum of the average cost (product of the average reliability) of invoked services. The expression for $E^k$ i.e., the mean execution time for class-$k$ requests, is evaluated under the hypothesis of the BCMP theorem [9][1].

In general, the Composed WS Optimizator has to mediate among multiple QoS attributes, which can be either mutually independent or possibly conflicting (e.g., usually the lower is the response time and the higher is the cost); therefore, the optimal service selection results in a multi-objective optimization. As in other approaches proposed in the literature [3,11,27], the multi-objective problem is transformed into a single objective problem by applying the Simple Additive Weighting (SAW) technique [16], one of the most widely used techniques to obtain a *score* from a list of dimensions. Since quality dimensions have different units of measure, the SAW method first normalizes the raw values for each quality dimension and then sums up the normalized values by considering the QoS weights. In this way the overall score associated with class $k$ can be evaluated as (see [2] for further details):

$$F^k(\mathbf{x}) = w_e^k \frac{E^k(\mathbf{x}) - E_{min}^k}{E_{max} - E_{min}} + w_c^k \frac{C^k(\mathbf{x}) - C_{min}^k}{C_{max}^k - C_{min}^k} + w_r^k \frac{R_{max}^k - R^k(\mathbf{x})}{R_{max}^k - R_{min}^k} \tag{6}$$

$E_{max}^k$ ($E_{min}^k$), $C_{max}^k$ ($C_{min}^k$) and $R_{max}^k$ ($R_{min}^k$) denote respectively the maximum (minimum) value for the execution time, the cost, and the reliability for class $k$. We will explain how to determine them after introducing the constraints of the optimization problem.

The objective function of the optimization problem is obtained by considering the weighted average of functions $F^k(\mathbf{x})$, where each function is weighted by the relative importance of class-$k$, $\Omega^k$ multiplied by the incoming workload $\gamma^k$:

$$F(\mathbf{x}) = \frac{\sum_{k \in K} \Omega^k \gamma^k F^k(\mathbf{x})}{\sum_{k \in K} \gamma^k} \tag{7}$$

[1] If the conditions of the BCMP theorem do not hold, (3) can be still used as a measure of the congestion at the concrete service $j$, that can be used to avoid highly congested nodes [8].

The service selection problem can be modelled by the following non-linear problem:

P1) $$max : F(\mathbf{x}) = \frac{\sum_{k\in K} \Omega^k \gamma^k F^k(\mathbf{x})}{\sum_{k\in K} \gamma^k}$$

$$\sum_{j\in J_i} x_{ij}^k = 1 \;\; \forall k \in K, i \in \mathcal{V}^k \tag{8}$$

$$\sum_{h\in K} \sum_{a\in \mathcal{V}^h} \frac{x_{aj}^h \lambda_a^h}{\mu_j} < 1 \;\; \forall j \in \mathcal{J} \tag{9}$$

$$E^k(\mathbf{x}) \le e_{max}^k \;\; \forall k \in K \tag{10}$$

$$C^k(\mathbf{x}) \le c_{max}^k \;\; \forall k \in K \tag{11}$$

$$R^k(\mathbf{x}) \ge r_{min}^k \;\; \forall k \in K \tag{12}$$

$$0 \le x_{ij}^k \le 1 \;\; \forall k \in K, i \in \mathcal{V}^k, j \in J_i^k$$

Constraints family (8) guarantees that all abstract service invocations are served by concrete services. Constraints family (9) entails the equilibrium condition for M/M/1 queue, while constraints families (10-12) are the global QoS constraints.

The values $E_{max}^k$, $C_{max}^k$, $A_{max}^k$, $E_{min}^k$, $C_{min}^k$, $R_{min}^k$ included in each function $F^k(\mathbf{x})$ can be determined as follows. $E_{max}^k$, $C_{max}^k$, and $R_{min}^k$ are set equal to the global constraints:

$$E_{max}^k = e_{max}^k;\; C_{max}^k = c_{max}^k;\; R_{min}^k = r_{min}^k \forall k \in K$$

$E_{min}^k$ is the minimum execution time which can be experienced by class-$k$ requests. For each abstract service $i \in \mathcal{V}^k$ the minimum execution time is given by $e_i^k(min) = \min_{j\in J_i^k} \frac{1}{\mu_j}$ (i.e., under the assumption that each concrete service $ws_j$ is under light load and hence its response time is estimated by its service time $\frac{1}{\mu_j}$). $E_{min}^k$ can then be evaluated by looking for the minimum cost path from the sink to the tank node of the composed service DAG, where $e_i^k(min)$ is considered as node cost. The same arguments hold for the evaluation of $C_{min}^k$ and $R_{max}^k$ where nodes cost (reliability) is set equal to $c_i^k(min) = \min_{j\in J_i^k} c_{ij}^k$ ($r_i^k(max) = \max_{j\in J_i^k} r_{ij}^k$).

Problem P1) is a non linear optimization problem in the continuous variables $x_{ij}^k$. In [2] we have shown that the objective function of problem P1) is neither concave nor convex and we have proposed a heuristic algorithm that gives good solutions for problem instances up to 300 abstract services and 100,000 candidates concrete services which can be solved in less than half an hour by a Pentium D workstation.

In the optimum solution of our *reference example*, the abstract services of the gold class are executed by the most reliable candidate services ($ws_6$, $ws_9$, $ws_5$, and $ws_{11}$), i.e., the assignment is deterministic. The overall reliability is equal to 0.988 and the total cost is 3.8. For the silver class, the first abstract service is executed by $ws_2$ which is the fastest server available, the node $i = 3$ is executed by $ws_5$, while the macro node $i = 2$ is executed with a probability $x_{23}^1$=0.7 by the fastest concrete service available ($ws_3$) and with a probability $x_{24}^1 = 0.3$ by service $ws_4$, with a total cost equal to the global constraint 3$.

## 6 Composed Service Quality Analyzer

The output provided by the composed WS optimizator is the set of variables $x_{ij}^k$ which determines the probability that the concrete service $ws_j$ will be invoked by the class-$k$ request when the workflow reaches the stage indicated by the macro-node $i$. The aggregated value of QoS for the abstract service $i$ then can be computed as the average of the quality dimensions of the invoked services weighted by $x_{ij}^k$. In this way, each abstract service can be considered as a black-box entity. The composed service quality analyzer derives models suitable for applying probabilistic model checking techniques. Software architects may exploit this prediction to evaluate and compare different alternatives at design-time.

Our approach starts from the composed services specifications and derives quality predictions, such as the composed service *Success probability* or *Mean Response Time* through the probabilistic model checker PRISM [15,22].

The software architect analyzes the output produced by the probabilistic model checker to verify if the service composition matches the quality goals required by the application domain. If these goals are not met, alternative compositions should be evaluated in order to reach the required goals (e.g., eliminating some concrete services candidates and/or changing global constraints).

The evaluation starts by translating the DAG representation to a Markov model. Depending on the nature of the composed service specification and on the type of analysis to be performed, different Markovian models can be chosen as output of the translation process.

In particular the Discrete Time Markov Chains (DTMC), and Continuous Time Markov Chains (CTMC) models can be considered. In our framework, the DTMC model is used to model simple service compositions when the average execution time is not considered in the analysis. A CTMC model is instead adopted if the analysis focuses on the average execution time. By modeling the transition probability as an exponential distribution, each service invocation can be represented as a state whose transition parameter is related to the expected duration of the service execution. Using a parameter $\Lambda$ representing the rate of the exponential distribution and defining it as $1/exeTime_i$ (see equation 2), the model approximates the real temporal behavior of the system, giving a time-depending probabilistic result. The system is characterized by an initial transient phase and finally probability values asymptotically stabilize.

We can analyze the model by verifying properties specified in temporal logic and evaluated through model checking. Basic properties on a service composition can be the reliability value of the whole complex system (e.g., the probability that starting from the initial state the system eventually reaches the success state), specified in PCTL as

$$P[F(system\_state = success)]$$

which states the probability that, *eventually* (operator $F$) the success state will be reached. Similar properties can be evaluated starting from each state of the system:

$$system\_state = \text{"a certain service invocation"} \Rightarrow P[F(system\_state = success)]$$

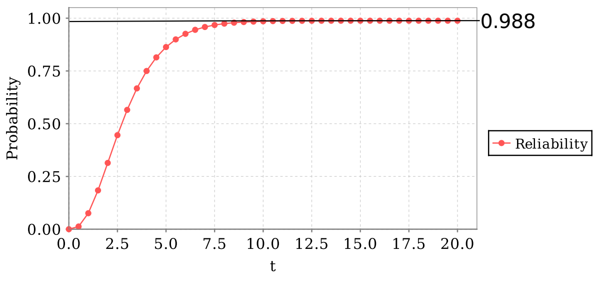


**Fig. 4.** Success probability evolution

The evaluation of these properties support the discovery of configurations that can be critical for the system. Properties can also be specified to obtain a boolean result. Indeed, we can also express properties like

$$P_{\geq threshold}[F(system\_state = success)]$$

whose evaluation yields a boolean value (true if the probability result complies with the threshold bound). Depending on the desired analysis, different logic properties can be formulated over the model and then submitted to the model checker.

The translation process from DAGs to Markov models is based on the exploration of the original model, starting from the initial node along the execution path defined by the control flow. The initial and final nodes of the DAGs correspond to the initial and final states of the Markov model. Each abstract service in a DAG is translated into a node with two outgoing transitions: the success transition and the failure transition. The probabilities associated with the two transitions depend on the annotations of the original composed service specification; the destination states can be the next state, in case of success, and a retry or a fail state, in case of failure. In particular, the Service Invocations Attempts (see Section 2) is used to determine the probability to reach the fail state of each abstract service. Details can be found in [15].

In our *reference example*, the composed WS quality analyzer determines an average execution time for the silver class equal to 17.22 sec. Figure 4 shows, as an example, the probability value of reaching the success state within time $t$ for the class gold process, computed for $t$ ranging in an interval $[0, 20]$ (seconds). In this case the aggregated values of QoS of each abstract service coincide with the corresponding values of the concrete services selected for the execution since the assignment is deterministic.

The plot in Figure 4 represents how the probability of success evolves over time after the invocation of the composed service. This value tends in the long run to the value obtained with the DTMC model (0.988), the reliability value of the service.

## 7 Related Work

Recently, QoS evaluations of Web service compositions have attracted great interest in the research community. Research related to our work can be classified into two main areas: (i) *QoS-based service selection*, and (ii) *probabilistic model checking for SOA*. In the *QoS-based service selection* research community, a first class of works provides some methods to derive performance related measures of workflow processes [12,18,23]. Cardoso [12] proposes two different metrics to evaluate the control-flow complexity of *BPEL* Web processes before their actual implementation. Dynamic Web service composition techniques can be classified into two main categories: composition by planning and business process optimization [25]. The former approach, proposed by the Semantic Web and AI communities, investigates the problem of synthesizing a complex behavior from an explicit goal and a set of candidate services which contribute to a partial solution of the complex problem. In the latter case [21,27], complex applications are specified as BPEL processes and the best set of services are dynamically selected at run time by solving an optimization problem. The Semantic Web and AI approach is very flexible since a composed service process is built automatically or semi-automatically from a high level specification of the required functionality. Anyway composed process synthesis is computationally intensive and QoS optimization is not the primary concern. Early solutions for business process optimization considered only *local constraints* (i.e., constraints which can pose restrictions only on the execution of individual abstract services). In that case, the service selection is very simple and can be performed at run time by a greedy approach which selects the best candidate service suitable for the execution [17]. More recent solutions support also *global constraints* [3,10,11,26,27] and are based on mixed integer linear programming, genetic algorithm or heuristics. In this work we have extended these solutions by considering the optimization of multiple process instances and evaluating Web service performance. Taking into account explicitly Web service response time makes the problem non-linear. With respect to the work in [11], we consider processes with different specifications; the optimization problem in this way becomes much harder since the objective function is non-convex.

The work in our group on *Probabilistic Model Checking for SOA* area is an evolution of previous work described in [5], which dealt with model checking service compositions described by BPEL orchestrations. This earlier approach did not refer to a probabilistic model checker, but the Bogor model checker [14] to perform design-time analysis was used.

## 8 Conclusions and Future Work

In this paper we have presented a model-driven approach, which automatically selects the set of services to be invoked at run time, transforms a design model of service composition into an analysis model, which then feeds a probabilistic model checker for quality prediction. Our framework is supported by a tool and our initial assessment of the approach has been encouraging. As future work, we will extend the optimization model in order to include abstract services parallel execution and to support deterministic execution as an explicit constraint. A further development of case studies will be also part of our future activities; specifically, we intend to take advantage from the case studies developed within the Q-ImPrESS and S-Cube projects to validate the proposed framework. Finally, we will also focus on systematizing the feedback loop from run-time observations of the quality attributes of a running composite service back to the design environment. If the running system is found to behave inconsistently with respect to the design model, it would be desirable to evaluate different architectural scenarios and possibly derive an improved implementation.

## Acknowledgments

This work has been partially supported by the project Q-ImPrESS and S-Cube NoE funded under the European Union's Seventh Framework Programme (FP7).

## References

1. Alves, A., et al.: Web service business process execution language version 2.0. Committee Draft (May 17, 2006)
2. Ardagna, D., Mirandola, R.: Service Selection Policies for the execution of Autonomic Services, note = Politecnico di Milano, Dipartimento di Elettronica e Informazione Technical report number 2008.13 (July 2008)
3. Ardagna, D., Pernici, B.: Adaptive service composition in flexible processes. IEEE Trans. on Software Engineering (June 2007)
4. Atkinson, C., Kuhne, T.: Model-driven development: A metamodeling foundation. IEEE Software 20(5), 36–41 (2003)
5. Baresi, L., Bianculli, D., Ghezzi, C., Guinea, S., Spoletini, P.: Validation of web service compositions. IET Software 1(6), 219–232 (2007)
6. Baresi, L., Gerosa, G., Ghezzi, C., Mottola, L.: Playing with time in publish-subscribe using a domain-specific model checker. In: SAVCBS 2007: Proceedings of the 2007 conference on Specification and verification of component-based systems, pp. 55–62. ACM, New York (2007)
7. Baresi, L., Ghezzi, C., Mottola, L.: On accurate automatic verification of publish-subscribe architectures. In: ICSE 2007: Proceedings of the 29th International Conference on Software Engineering, pp. 199–208. IEEE Computer Society, Washington (2007)
8. Bertsekas, D., Gallager, R.: Data Networks, 2nd edn. Prentice Hall, Englewood Cliffs (1991)
9. Bolch, G., Greiner, S., de Meer, H., Trivedi, K.S.: Queueing Networks and Markov Chains. J. Wiley, Chichester (1998)
10. Canfora, G., di Penta, M., Esposito, R., Villani, M.L.: QoS-Aware Replanning of Composite Web Services. In: ICWS 2005 Proc., Orlando (2005)
11. Cardellini, V., Casalicchio, E., Grassi, V., Mirandola, R.: A framework for optimal service selection in broker-based architectures with multiple QoS classes. In: Services computing workshops, SCW 2006, pp. 105–112. IEEE computer society, Los Alamitos (2006)
12. Cardoso, J.: Complexity analysis of bpel web processes. Software Process: Improvement and Practice 12(1), 35–49 (2007)
13. Chandrasekaran, S., Miller, J.A., Silver, G., Arpinar, I.B., Sheth, A.P.: Performance Analysis and Simulation of Composite Web Services. Electronic Market: The Intl. Journal of Electronic Commerce and Business Media 13(2), 120–132 (2003)
14. Dwyer, M.B., Hatcliff, J., Hoosier, M., Robby.: Building your own software model checker using the bogor extensible model checking framework. In: Etessami, K., Rajamani, S.K. (eds.) CAV 2005. LNCS, vol. 3576, pp. 148–152. Springer, Heidelberg (2005)
15. Gallotti, S., Ghezzi, C., Mirandola, R., Tamburrelli, G.: Quality prediction of service compositions through probabilistic model checking (2008)

16. Hwang, C.L., Yoon, K.: Multiple Criteria Decision Making. Lecture Notes in Economics and Mathematical Systems. Springer, Heidelberg (1981)
17. Maamar, Z., Sheng, Q.Z., Benatallah, B.: Interleaving web services composition and execution using software agents and delegation. In: WSABE 2003, Melbourne (2003)
18. Marzolla, M., Mirandola, R.: Performance prediction of web service workflows. In: Overhage, S., Szyperski, C., Reussner, R., Stafford, J.A. (eds.) QoSA 2007. LNCS, vol. 4880, pp. 127–144. Springer, Heidelberg (2008)
19. Menascé, D.A., Dubey, V.: Utility-based qos brokering in service oriented architectures. In: ICWS (2007)
20. Ouzzani, M., Bouguettaya, A.: Efficient Access to Web Services. IEEE Internet Comp. 37(3), 34–44 (2004)
21. Patil, A.A., Oundhakar, S.A., Sheth, A.P., Verma, K.: METEOR-S web service annotation framework. In: WWW 2004 Proc., New York, pp. 553–562 (2004)
22. PRISM, Probabilistic Model Checker, http://www.prismmodelchecker.org/
23. Rud, D., Schmietendorf, A., Dumke, R.: Performance modeling of ws-bpel-based web service compositions. Scw 0, 140–147 (2006)
24. SNOPT, Software for Large-Scale Nonlinear Programming
25. Srivastava, B., Koehler, J.: Web service composition — current solutions and open problems. In: ICAPS 2003 Proc. (2003)
26. Yu, T., Zhang, Y., Lin, K.-J.: Efficient algorithms for web services selection with end-to-end qos constraints. ACM Trans. Web 1(1), 1–26 (2007)
27. Zeng, L., Benatallah, B., Dumas, M., Kalagnamam, J., Chang, H.: QoS-aware middleware for web services composition. IEEE Trans. on Software Engineering 30(5) (May 2004)

# Semantic-Aware Service Quality Negotiation[⋆]

Marco Comuzzi[1], Kyriakos Kritikos[2], and Pierluigi Plebani[1]

[1] Politecnico di Milano – Dipartimento di Elettronica e Informazione
Piazza Leonardo da Vinci 32, 20133 Milano, Italy
{comuzzi, plebani}@elet.polimi.it
[2] Institute of Computer Science, FORTH
Heraklion, Crete, Greece
kritikos@ics.forth.gr

**Abstract.** The goal of Web service (WS) discovery is to select WSs that satisfy both the users' functional and non functional requirements. Focusing on non functional requirements, a matchmaking algorithm usually takes place to verify if the quality offered by the WS provider overlaps the quality requested by the user. Since quality, in a provider perspective, is costly, a further step, a negotiation, should be performed to identify a mutually agreed quality level. In this work, we join previous work on a semantic-based quality definition model and WS negotiation, to provide a framework enabling semantic-aware automated WS negotiation. More specifically, OWL-Q, a semantic QoS-based WS description language, is extended with appropriate negotiation concepts and properties.

## 1 Introduction

According to the OASIS (http://www.oasis-open.org) definition the "Service Oriented Architecture (SOA) is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains". Thus, the ownership holds a key-role in realizing a SOA: for instance, who builds and makes a service available might be different from who consumes the service and they might not know each other in advance. As a consequence, at design-time, service providers try to identify which could be the requirements of a potential user and develop the service accordingly. On the other side, service consumers need to select the best service among a set of available ones, considering both *what* the service does (functional perspective), and *how* the service works (non-functional perspective).

In this paper, we focus on the non-functional perspective and, in particular, on the quality of service (QoS hereafter) negotiation, i.e., the process that produces an agreement between a service consumer and a service provider with respect to a) the QoS a service must ensure during its execution and b) the amount of money the consumer has to pay. In this scenario, the consumer and the provider

[⋆] 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) and the Italian FIRB Project TEKNE.

© Springer-Verlag Berlin Heidelberg 2008

might be aware of each other just before the service invocation takes place. Thus, the research direction is to make the negotiation process as much automatic as possible [11], but this goal is achievable only if both service providers and users agree on the same model for expressing QoS dimensions.

In a previous work [1], we introduced an approach where as soon as the users expectations are defined and providers clarify their capabilities, then the matchmaking and the agreement processes are automatically performed on-the-fly. The main limitation of this approach is about the assumption that both users and providers must share the same set of quality dimensions to describe the requirements and capabilities, respectively. Actually, in some application domains, different domain experts may not come to a complete agreement for a common (domain-dependent) QoS model. In this way, there may be two or more 'standard' QoS models for a specific application domain and users may choose one of them for expressing their QoS requirements/capabilities.

In this paper, we aim at overcoming this limitation by adopting and improving OWL-Q [5], an extension of OWL-S for a rich, semantic, and extensible QoS-based service description. There are lot of reasons for using ontologies and rules to express quality models and to automate QoS matchmaking and negotiation. First, ontologies provide a formal, syntactic, and semantic description model of concepts, properties and relationships between concepts. They give meaning to concepts like QoS dimensions, value types, offers, and requests so that they are human-understandable and machine-interpretable, while providing the means for interoperability. Moreover, ontologies are extensible as new concepts, properties, or relationships can be added to an ontology. In addition, Semantic Web (SW) techniques can be used for reasoning about concepts or for ontology mapping. These techniques can lead to the syntactic and semantic matching of ontological concepts and to the enforcement of class and property (e.g. type checking, cardinality) constraints. Therefore, by providing semantic description of concepts and by supporting reasoning mechanisms, ontologies cater for better discovery process with higher precision and recall. Last but not least, ontologies can help specialized agents in performing very complex reasoning tasks like service discovery, mediation, or negotiation. If QoS models and specifications are expressed with ontologies, then they could be aligned with each other so that the WS discovery and negotiation processes are not actually affected by a pre-existent ontology mismatch. This alignment process is realized with the additional use of rules that define mappings between concepts of different models and specifications.

The discussion about our proposal on semantic-aware quality negotiation will be tied to a running example, i.e. *SMS Monitoring*. This service is available to all the users that have a contract with a national mobile phone company. When a user sends to the service his or her mobile phone number, the service returns how many SMSs have been sent from that number starting from the beginning of the current month. In this case, we consider service *availability*, *response time*, and *coverage* (i.e., how many mobile phone companies can be queried by the service) as the most relevant quality dimensions.

The work is structured as follows. In Section 2, we introduce the requirements of a quality model that need to be satisfied to support automated negotiation. Section 3 briefly analyzes the main elements composing OWL-Q. In Section 4, we extend OWL-Q with new concepts specifically intended for supporting automated negotiation and we introduce examples of rules to reason about QoS matching and negotiation. Finally, Section 5 discusses related work, while Section 6 concludes the paper outlining possible future research directions.

## 2 Quality Model Requirements

As discussed in Kritikos and Plexousakis [6], we can define a set of requirements that a WS QoS description needs to satisfy to be adopted. First, quality can be defined both as a *requirement* and as a *capability*. Thus, it should be possible to specify both the QoS properties that clients require and the QoS properties that services provide. These two aspects should be specified in two separated documents that must be compared during the service discovery phase, to realize if the quality provided by a service satisfies the user's requirements.

Due to the high dependability of quality definition on the application domain, a QoS model has to be *extensible*, that is, it has to include both domain independent QoS dimensions, and domain specific QoS dimensions. Moreover, new domain specific criteria could be added and used to evaluate QoS without changing the underlying computation (i.e. matchmaking and ranking) model. In order to allow knowledge sharing and the comparison between capabilities and requirements, users and providers have to agree on the adopted syntax and semantics. About the syntax, the QoS model has to be *compliant* with already widely-accepted standards, e.g., WS-Policy. Concerning semantics, QoS concepts must be *formally* described in order to have terms/concepts with specific meaning for both requesters and providers.

To improve its flexibility, a QoS model should be *syntactically separated* from other parts of service specification, such as the interface definition. On the one hand, this improves reusability in describing several services with the same QoS or a service with different levels of QoS. On the other hand, this allows the specification of *classes of service*, determined by the discrete variation of the complete service and QoS provided by one WS.

Due to these initial requirements, QoS models are usually defined by a composition of several quality dimensions (a.k.a. quality parameters, or quality attributes). Each attribute is measured with the help of a *metric* that gives an objective way to state which are the possible and actual values for a given dimension. Quality dimensions are important inputs to the overall QoS of a service. Some attributes are common across domains and some are specific to domains. More specifically, a *QoS dimension* should be defined at least by the following aspects:

*(i)* The value set for the metric (and its allowed value range) to determine which are the admissible values for the dimension;

*(ii)* The domains that this attribute belongs to;

*(iii)* The weight of the metric relative to its domain and user preferences (to rank the dimensions in order of importance);

*(iv)* The characteristic of the function, from metric values to overall QoS values, to determine how the quality varies with respect to variation of a quality dimension;

*(v)* The temporal characteristic of the metric value;

*(vi)* The description (mathematical or otherwise formal) of how a QoS metric value of a complex WS can be derived from the corresponding QoS metrics values of the individual WSs that constitute the complex one;

*(vii)* A set of reference ontologies, e.g., ontology of measurement units, ontology of currency units, ontology of measured properties and ontology of measurement methods.

The first aspect, i.e., the value set for the metric, represents the starting point for the matchmaking phase. Indeed, in the user requirements document, this set expresses the value range in which a quality dimension may vary during the service execution. For instance, the user may request $availability\_r \in [95\%..99\%]$ [1]. On the other side, in the provider capabilities document, this set expresses the value range in which the provider promises that the quality dimension varies (e.g., $availability\_c \in [90\%..99\%]$).

The intersection between these two sets is evaluated during the matchmaking phase to state if a non empty set of values exists, which the provider supports and that satisfies the user, e.g., $availability\_r \cap availability\_c \in [95\%..99\%]$. If this happens for all the quality dimensions included in the requirements document, then the provider is able to support all the user requirements. It is worth noting that the matchmaking only gives a technical evaluation, i.e., it states that the user requirements may be fulfilled, even though an economical evaluation must be performed in order to determine if the actual values assumed by a quality dimension can be supported. For this reason, we also consider the negotiation as a further step after the matchmaking. The goal of the negotiation is to identify, for each quality dimension, which are the values that maximize the expectations of the user who has a specific budget. As a consequence, the QoS model has to include elements for evaluating the cost for supporting (provider perspective) or receiving (user perspective) a given quality dimension level. Since in this paper we also deal with negotiation, the requirements of a QoS model introduced in [6] and discussed above have to be updated accordingly.

From the provider perspective, the QoS model needs to include the *cost model*, i.e., a function that calculates how much is the effort for the provider for offering a given value for a quality dimension. In this way, the provider calculates how much is the cost for providing a set of capabilities and, consequently, the provider decides the *price* for the service. For instance, the cost may be proportional to the availability value according to the following formula: $cost(availability) = availability \star 3\$$. Thus, assuming that the provider has a fixed revenue of 5$, the price for having an availability in the range [0.95%..0.97%] is ∈ [7.85$..7.97$].

[1] Hereafter, we use the characters 'r' and 'c' as subscripts to indicate a quality dimension in the request or capabilities document, respectively.

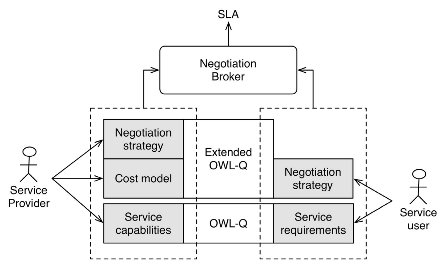


**Fig. 1.** Overall negotiation framework

From the user perspective, the QoS model needs to consider the user's *budget*, i.e., the amount of money the user is willing to pay for the service. During the negotiation the budget is strictly related to the weights that express the user preferences among the quality dimensions. Thus, the QoS model needs to consider the *Negotiation strategy*. Usually, the strategy assumes that the greater is the weight, the higher is the preference on that quality dimension. Thus, these weights indicate how the budget should be split among the quality dimensions. Assuming an overall budget of 20$, and availability and response time equally important (their related weights are both 0.5), then the user is willing to pay up to 10$ and, accordingly to the cost model expressed above, this can be feasible. On the contrary, assuming 7.9$ as the budget for availability, then the range of values for this quality dimension must be rearranged accordingly, i.e., [0.95%..0.966%].

It is worth noting that not all the quality dimensions can be negotiable. A user can ask that the range/set of values for a specific quality dimension must be entirely supported. For instance, for the user may be mandatory that coverage includes the values of Orange and Verizon. So, the negotiation strategy is not allowed to modify this value set.

On the basis of the above quality model requirements, in this work we propose a framework for semantic-aware negotiation. As shown in Figure 1, we rely on OWL-Q for expressing user capabilities and provider requirements. An extension of OWL-Q, discussed in Section 4, allows the definition of negotiation strategies and cost models that will be used by the negotiation broker to generate the SLAs.

## 3 OWL-Q

OWL-Q has been introduced by Kritikos and Plexousakis in [5] with the objective of providing a means for rich, semantic, and extensible QoS-based WS description. OWL-Q is an upper ontology that satisfies all the requirements introduced in the previous section. The OWL-Q ontology complements OWL-S [12] and comprises of many sub-ontologies/facets. Each facet concentrates on a particular aspect of the QoS modeling and can be extended independently of the others. It is worth noting that some of the concepts here presented, that are

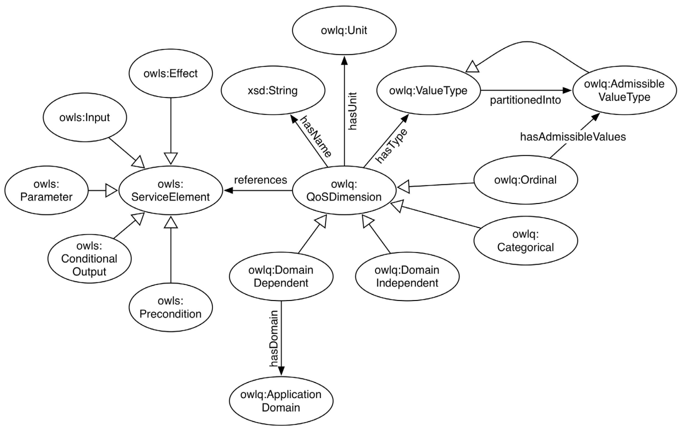


**Fig. 2.** Overview on OWL-Q

also discussed in depth in [5], have been revised according to the quality model presented in [1], which puts the basis for a model oriented towards WS QoS negotiation.

The main element in OWL-Q is the *QoSDimension* (see Figure 2) that can be attached to any *owls:ServiceElement* for expressing preferences or capabilities of a service element of any type. Each dimension has a *Name* and can assume a set of values of a given *ValueType*. A dimension can be either *Categorical* or *Ordinal*. An example of a *Categorical* dimension is *coverage*, where its set of values are the list of mobile phone operators supported by the service, e.g., *Orange, Verizon, Cingular*. In case of *Ordinal* dimensions, the value type is a range of values in which a quality dimension varies. For instance, *availability* can vary from [0%..100%]. This set can be partitioned in several sub-ranges, i.e.,*AdmissibleValueType*s. This allows both users and requesters to express the values of requirements or capabilities not as a single value, but as a range of permitted values (e.g., [90%..94%];[95%..99%]). An additional classification is given by the dependency on the application domain. Some quality dimensions are *Domain Independent*, so they can be useful regardless of the considered type of service (e.g., *response time* and *availability*); on the contrary, other quality dimensions are *Domain Dependent* and are related to an *Application domain* (e.g., *coverage*).

A definition of who is in charge of measuring a QoS dimension and in which way the measurement takes place is given by the *OWL-Q Metric facet* shown in Figure 3. In more detail, the values of a dimension are provided by a *Party* that is in charge of measurement. This actor might be a *Provider*, a *Requester*, or a *Third-party*. There are simple QoS metrics *measuredBy* a *MeasurementDirective* or complex ones. *ComplexMetric*s are derived from other metrics with the help of a *MetricFunction*. Metrics can be positively or negatively monotonic. In this way, we know if one metric's value is better than another value. This is fundamental whenever we have to compare several quality definitions in order to state which

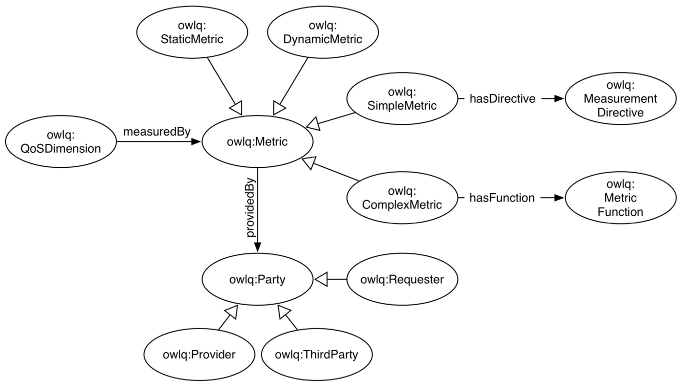


**Fig. 3.** OWL-Q Metric facet

is the best one. In addition, a *Metric* can be also classified as static or dynamic metric: a *StaticQoSMetric* is computed only once according to a trigger, whereas a *DynamicQoSMetric* is computed repeatedly according to a schedule.

## 4 Semantic-Aware Negotiation

In the multiagent computing literature, a generic automated negotiation framework is usually defined by three elements [3]:

**Negotiation Object.** It represents the features of what is under negotiation. In particular, the definition of the negotiation object concerns the identification of the issues under negotiation, their properties, and their admissible values;

**Negotiation Protocol.** It identifies the rules that must be followed by the negotiation participants, defining the admissible states of a negotiation and the behaviors that can be endorsed by participants. Classic negotiation protocols are the iterated bilateral negotiation protocol, in which two negotiators alternatively exchange offers, or price-only auction protocols, in which a third party, i.e., the auctioneer, collects offers from the negotiators and identifies the winner according to a pre-specified market clearing rule;

**Negotiators' decision model.** It identifies the strategy of the participants in the negotiation. In particular, the decision model sets the rules followed by a negotiator to generate new offers and to decide whether to accept or refuse offers, or to withdraw from the negotiation.

Our OWL-Q extensions focus on the Negotiation Object and on the Negotiators' decision models. We extend OWL-Q with a new set of concepts and properties, which are grouped as *Quality-related* and *Negotiation-specific* extensions. The former extend the ontology in order to accommodate the Negotiation Object, i.e., the description of what can be negotiated. The latter introduce new

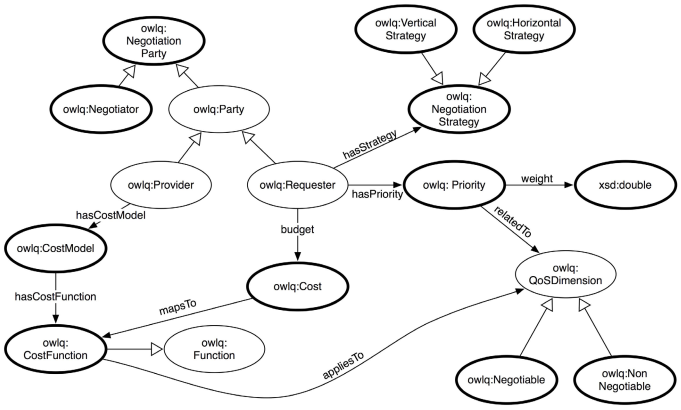


**Fig. 4.** OWL-Q Extensions

concepts, such as the negotiator actor or service consumers' negotiation strategies, which are then used to define the negotiators' decision model. Figure 4 reports the extended OWL-Q.

For what concerns the negotiation protocol, the extension of OWL-Q that describes various negotiation protocols, such as trading and tendering, is currently under development. In this paper, our approach is limited to QoS configuration algorithms for which the the negotiators' strategies are parameterized [1]. The WS QoS configuration protocols proposed in the paper can be implemented by a broker-based architecture for QoS negotiation, as shown in our framework in Figure 1.

We introduce quality-related extensions to cope with two fundamental issues raised by the need to support QoS negotiation through an ontology:

(i) not all QoS dimensions, either technical or domain dependent, are negotiable. A QoS dimension is *Negotiable* when its value can be set by the service provider at runtime, i.e., when the service is invoked. *Non-negotiable* QoS dimensions are the ones for which the value cannot be set at runtime by the service provider. For instance, when a service is invoked, its reputation is fixed, regardless of the technique adopted for assessing reputation. Moreover, the service provider can always decide whether to allow or not the negotiation of a specific quality dimension. In our SMS service running example, *response time* can be considered as negotiable, since the response time value can be altered by the provider at execution time according to the customer requirements. However, in case the provider's provisioning infrastructure does not allow such an adaptation, our ontology allows the provider to declare the response time as non negotiable;

(ii) In order to automatically negotiate the QoS of a service [1], we also need to define a total ordering relation among admissible values identified for each QoS dimension. This ordering is established by the communities of domain experts that define the quality documents associated to a category of WSs.

Negotiation-specific extensions concern the concepts and properties, besides QoS definition, required to establish a negotiation framework. In particular, we identify the following concepts:

**Negotiator Actor.** Usually, the participants involved in WS QoS negotiation are the service provider and the service consumer. However, a more flexible approach should consider that the execution of a negotiation may be delegated to a trusted third party, such as, for instance, an ad-hoc agent explicitly designed to negotiate on the behalf of the service provider or the service customer. Introducing the negotiation actor also enables our framework to accommodate other negotiation protocols, such as, for instance, single-text mediated negotiations, which require the existence of a third party trusted by all the negotiation participants;

**Negotiation Strategy.** As previously introduced, our semantic-aware negotiation framework relies on the assumption of having parameterized negotiation strategies, i.e., negotiation strategies that are fully determined when a negotiator actor specifies the values of a set of parameters, such as the initial offer and the degree of concession over time to the counterpart.

**Cost model.** From the provider's point of view, the negotiation often relies on a cost model, i.e., parameterized functions that are used to evaluate the cost sustained by the service provider for giving a certain level of quality in his or her service offers. Usually, WS QoS cost models are additive, that is, the cost of a service offer is given by the costs associated by the cost model to each single QoS value that appears in the offer.

Derivation of new knowledge is actually the main driving reason for using rules with ontologies. New knowledge may come in different forms, such as equivalence of quality dimensions, matchmaking of a QoS offer with a demand, producing the price of a specific QoS level for a provider, etc. Generally, the whole discovery and negotiation algorithms may be written in the form of a modular set of rules so that only specific functions need to be actually implemented in places where mathematical tools are required.

By using the application domain of SMS Monitoring, we are now going to show a small example of how reasoning can support the negotiation process. Assume that a WS provider advertises that his or her WS has *availability* in the range of $[0.9, 0.99]$ (value type is $[0.0, 1.0]$), *response time* in the range of $[0.5, 2.0]$ seconds (value type is $(0.0, 2.0]$) and $coverage = \{Orange, Verizon, Cingular\}$. In addition, let us assume that the cost model of the WS provider is defined by the formula: $price = cost_{avail} + cost_{resp} + cost_{cov}$ dollars, where $cost_{avail} = avail * 3$, $cost_{resp} = 3 * \frac{2-resp}{1.5}$ and $cost_{cov} = |coverage_{ad} \cap coverage_{req}|$. Further, let us also assume that there is a WS requester that requests a WS having *availability* in $[95, 99]\%$ (value type is $[0, 100]$), *response time* in $[100, 1000]$ milliseconds (value type is $(0, 2000]$) and $coverage = \{Orange, Verizon\}$. Finally, assume that the WS requester has budget 7\$.

Now consider that the WS's capabilities *ad* and the WS requester's requirements *req* have been submitted to a negotiation broker by using our proposed semantic QoS model. This broker uses the following rules (in abstract form) in order to infer if the QoS offer and demand are compatible for negotiation:

$$\begin{aligned}
&matches\,(ad, req) \Leftarrow \forall qdi_1 \in req\ \exists qdi_2 \in ad \text{ s.t } match\,(qdi_1, qdi_2)\\
&match\,(qdi_1, qdi_2) \Leftarrow equiv\,(qdi_1, qdi_2) \wedge c\_values\,(qdi_1, qdi_2)\\
&c\_values\,(qdi_1, qdi_2) \Leftarrow \exists v_1 \in qdi_1.values \wedge \exists v_2 \in dqi_2.values \text{ s.t}\\
&\qquad utf_{qdi_1.unit \mapsto qdi_2.unit}\,(v_1) = v_2\\
&compatible\,(ad, req) \Leftarrow matches\,(ad, req) \wedge req.Bugdet \geq MinCost\,(ad, req)
\end{aligned}$$

The first rule expresses that the QoS offer matches the QoS demand when for each quality dimension (qd) in the demand there exists a corresponding matching qd of the offer. The second rule expresses that two qds match if they are equivalent and they have a common value in their corresponding admissible values. Equivalence of qds is actually reduced to equivalency of their metrics. Kritikos and Plexousakis in [5] propose a semantic QoS metric matching algorithm in which two simple metrics are equivalent if they have compatible value types and units. In our example, considering the aforementioned algorithm, it is easy to see that the availability and response time of the *ad* and *req* specs are equivalent. More details of how this equivalence is inferred can be found in [5]. The third rule expresses that two qds have common values if there exists a value included in the admissible value type of the first qd that can be transformed to a value included in the admissible value type of the second qd by using a utility transformation function (utf). In our example, value 1000 of *req*'s response time admissible value type is transformed to value 1.0 which is included in the admissible value type of *ad*'s response time by using the utf: $utf_{ms \mapsto s}(x) = \frac{x}{1000}$. This is also the case for the availability qd (the case of coverage qd is trivial). Thus, based on the first three rules, one can infer that the QoS offer and demand of our example can be matched.

The fourth and final rule infers that a QoS offer *ad* is compatible to a QoS demand *req* if they match and the requester's budget is less or equal to the minimum cost of the WS. Rule *MinCost* can be a user function that transforms the offer and the demand into an optimization problem, in order to find the minimum cost of the WS that respects the constraints of the demand. Based on our example, the smallest common value is 0.95 for availability and 1.0 for response time. Therefore, the minimum cost of the WS will be: $2 + 3 * 0.95 + 3 * (2 - 1)/1.5 = 6.85$, which is less than the requester's budget. Thus, finally, the QoS offer and the demand are compatible as the fourth rule is satisfied.

## 5 Related Work

The need for automated management of quality SLAs or, more generally, contracts, is being addressed as one of the main driver for the adoption of service based systems in real-world scenarios [11]. As already remarked, we argue that giving formal semantics to the description of QoS and the elements that compose the negotiation framework represents a tenet for modern service based systems. Semantic-based negotiation mechanisms and protocols have been often inspired by the agent community literature (see [7] for a survey on approaches for multi-attribute negotiation in Artificial Intelligence).

Focusing on the SW community, Chiu et al. [2] discuss how ontology can be exploited for supporting negotiation. In particular, the authors highlight how shared and agreed ontologies provide common definitions of the terms to be used in the subsequent negotiation process. Lamparter et al. [8] introduce a model for specifying policies for automated negotiation of WSs, by relying on the upper ontology DOLCE [9]. About the use of ontologies for specifying the agreement among parties, [10] presenta reasoning methods for the components of a WS-Agreement agreement which must be compatible for quality matches. With the same goal [13] discusses the KAoS policy ontology, which allows the specification, management, and enforcement of policies within the specific contexts established by complex organizational structures. Other approaches, such as [14], focus only on semantically describing the QoS capabilities and requirements of WSs for the purposes of WS discovery.

onQoS-QL [4] and OWL-Q [5] are the most rich semantic languages for QoS-based WS description adopting all the requirements expressed in [6]. They are also supported by WS discovery frameworks. However, the main drawback of onQoS-QL is that its expressivity concerning metric functions, directives, and QoS constraints is limited. In addition, the QoS profile of a WS contains only QoS metrics and not QoS constraints on these metrics.

Based on the above analysis, it is clear that each research approach, independently of its efficiencies and deficiencies, describes QoS for WSs focusing on supporting either WS discovery or WS negotiation. So in order to support both of the latter two processes, one has to follow two alternative directions: either use the best approach in each process and provide a mapping between them or create a new uniform approach by extending an existing one. This paper follows the second direction as it seems more promising and efficient, extending one of the best approaches for QoS-based WS description and discovery in order to further support WS negotiation.

## 6 Conclusion

Although the need for a semantic-aware description of the quality of WS is now recognized, most of the current work mainly focus on the definition of quality attributes. In this paper, we have proposed to go one step ahead discussing the need for a semantic-oriented negotiation in SOA. With this aim, starting from an existing QoS ontology, i.e., OWL-Q, we have identified which are the missing elements and we have proposed possible extensions that allow to deal with the negotiation process. As our work is preliminary, it can be further extended in terms of concepts and mechanisms for reasoning on the quality attributes to assist and automate the algorithm for enacting the negotiation.

## References

1. Cappiello, C., Comuzzi, M., Plebani, P.: On automated generation of web service level agreements. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) CAiSE 2007 and WES 2007. LNCS, vol. 4495, pp. 264–278. Springer, Heidelberg (2007)
2. Chiu, D.K.W., Cheung, S.C., Hung, P.C.K., Fung Leung, H.: Facilitating e-negotiation processes with semantic web technologies. In: Proc. 38th Annual Hawaii International Conference on System Sciences (2005)
3. Faratin, P., Sierra, C., Jennings, N.R.: Negotiation decision functions for autonomous agents. Int. Journal of Robotics and Autonomous Systems 24(3-4), 159–182 (1998)
4. Giallonardo, E., Zimeo, E.: More semantics in qos matching. In: Int. Conf. on Service-Oriented Computing and Applications, pp. 163–171 (2007)
5. Kritikos, K., Plexousakis, D.: Semantic qos metric matching. In: Proc. of ECOWS 2006, pp. 265–274 (2006)
6. Kritikos, K., Plexousakis, D.: Requirements for qos-based web service description and discovery. In: Proc. of COMPSAC 2007, pp. 467–472 (2007)
7. Lai, G., Li, C., Sycara, K., Giampapa, J.A.: Literature review on multi-attribute negotiations. Technical Report CMU-RI-TR-04-66, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA (December 2004)
8. Lamparter, S., Luckner, S., Mutschler, S.: Formal specification of web service contracts for automated contracting and monitoring. In: Proc. HICSS 2007 (2007)
9. Masolo, C., Borgo, S., Gangemi, A., Guarino, N., Oltramari, A., Schneider, L.: Wonderweb deliverable d17. the wonderweb library of foundational ontologies and the dolce ontology
10. Oldham, N., Verma, K., Sheth, A., Hakimpour, F.: Semantic ws-agreement partner selection. In: WWW 2006, pp. 697–706 (2006)
11. Papazoglou, M., Traverso, P., Dustdar, S., Leymann, F.: Service-Oriented Computing: State of the art and research challenges. IEEE Computer 11, 38–45 (2007)
12. Sycara, K., et al.: OWL-S 1.0 Release. OWL-S Coalition (2003), http://www.daml.org/services/owl-s/1.0/
13. Uszok, A., Bradshaw, J., Jeffers, R., Suri, N., Hayes, P., Breedy, M., Bunch, L., Johnson, M., Kulkarni, S., Lott, J.: Kaos policy and domain services: Toward a description-logic approach to policy representation, deconfliction, and enforcement. In: Proc. 4th IEEE Int. Workshop on Policies for Distributed Systems and Networks (2003)
14. Zhou, C., Chia, L.-T., Lee, B.-S.: Daml-qos ontology for web services. In: Proc. IEEE ICWS 2004, pp. 472–479 (2004)

# Multi-level SLA Management for Service-Oriented Infrastructures*

Wolfgang Theilmann[1,**], Ramin Yahyapour[2,**], and Joe Butler[3,**]

[1] SAP Research, Vincenz-Priessnitz-Str. 1, 76131 Karlsruhe, Germany
wolfgang.theilmann@sap.com
[2] Dortmund University of Technology, ITMC, 44221 Dortmund, Germany
ramin.yahyapour@udo.edu
[3] Intel Ireland Ltd., Collinstown Industrial Estate, Leixlip, Ireland
joe.m.butler@intel.com

**Abstract.** The ongoing transformation of a product-oriented economy towards a service-oriented economy has come to a critical point. In order to have services as tradable goods, the conditions of their provisioning need to be exactly specified and managed. Service Level Agreements (SLAs) have become a common means for specifying these conditions at a singular level. However, realistic service provisioning scenarios involve multiple stakeholders and layers of a business/IT stack.

This paper presents an approach for multi-level SLA management, where SLAs are consistently specified and managed within a service-oriented infrastructure (SOI). We present the general approach of an SLA management framework, a conceptual architecture and some insights into industrial practice in various domains.

**Keywords:** service level agreement (SLA), service-oriented infrastructure (SOI), e-contracting, adaptive infrastructures, manageability, non-functional properties.

## 1 Introduction

The ongoing transformation of a product-oriented economy towards a service-oriented economy has come to a critical point. IT-supported service provisioning has become of major relevance in all industries and domains. However, the nature of these setups is typically quite static because it requires significant effort to create service offers, to negotiate provisioning details with customers and to manage and control provided services.

* The research leading to these results is partially supported by the European Community's Seventh Framework Programme ([FP7/2001-2013]) under grant agreement n° 216556.

** On behalf of the SLA@SOI consortium [3] including T. Ellahi, H. Li (SAP), F. Torelli (Engineering), J. Kennedy (Intel), M. Alvarez, A. Castro, J. Lambea (Telefonica Investigación y Desarrollo), C. Kotsokalis (University of Dortmund), M. Trifu, C. Momm (Research Centre Karlsruhe), A. Marconi, M. Pistore (Fondazione Bruno Kessler), L. Baresi (Politechnico Milano), G. Spanoudakis (CITY University), R. Perrot, T. Harmer (Queens University Belfast), G. Pipan (XLAB), G. Armellin (GPI) and M. Evenson (eTel).

© Springer-Verlag Berlin Heidelberg 2008

Truly dynamic service provisioning will be a major milestone for the further evolution towards a service-oriented economy, where IT-based services can be flexibly traded as economic good, i.e. under well defined and dependable conditions and with clearly associated costs. Eventually, this will allow for dynamic value networks that can be flexibly instantiated thus driving innovation and competitiveness.

Service Level Agreements (SLAs) have become a common means for specifying the conditions under which a certain service is provisioned by a service provider to a service consumer. However, SLA management frameworks typically focus at the level of singular service interfaces and do not recognize/support the fact that many services are composed on lower-level services, might involve 3rd party service providers and rely on a possibly complex business/IT stack [1].

In order to realize this vision of dynamic service provisioning we see 3 main challenges (also identified in [2]):

- *Predictability & Dependability*: The quality characteristics of services must be predictable and enforceable at run-time.
- *Transparent SLA management*: Service level agreements (SLAs) defining the exact conditions under which services are provided/consumed must be transparently managed across the whole business and IT stack.
- *Automation*: The whole process of negotiating SLAs and provisioning, delivery and monitoring of services must be automated allowing for highly dynamic and scalable service consumption.

The following business scenario highlights the integrated view on these challenges: A service provider offers services with differentiated, dependable and adjustable SLAs and can negotiate concrete SLAs with (individual or groups of) customers in an automated fashion. This business goal imposes requirements on software providers (to provide components with predictable non-functional behaviour) and infrastructure providers (to support an SLA aware management of resources) and also the service provider (to translate and manage SLAs from business level along the IT stack down to the infrastructure). Of course, complete business value chains can be easily composed on top of this setup.

The research project SLA@SOI [3] addresses the integrated research on the aforementioned challenges. The main goal is to provide an SLA management framework that allows for consistent specification and management of SLAs in a multi level environment. The framework is designed for integration into different service-oriented infrastructures and will be evaluated within various complementary industrial use cases.

The remainder of this paper is organized as follows. Section 2 introduces the technical approach for the SLA management framework and its most important elements. Section 3 discusses the scientific state of the art. Conceptual target architecture is then defined in Section 4 while Section 5 highlights current state of practice in selected industrial areas. Section 6 concludes with a brief summary and outlook.

## 2 SLA Management Framework

The objective of the SLA Management Framework is to support a holistic view for the management of service level agreements (SLAs) and to implement a framework that can be easily integrated into a service-oriented infrastructure (SOI). The main innovative features of the framework are (1) an automated e-contracting framework, (2) systematic grounding of SLAs from the business level down to the infrastructure, (3) exploitation of virtualization technologies at infrastructure level for SLA enforcement, and (4) advanced engineering methodologies for creation of predictable and manageable services.

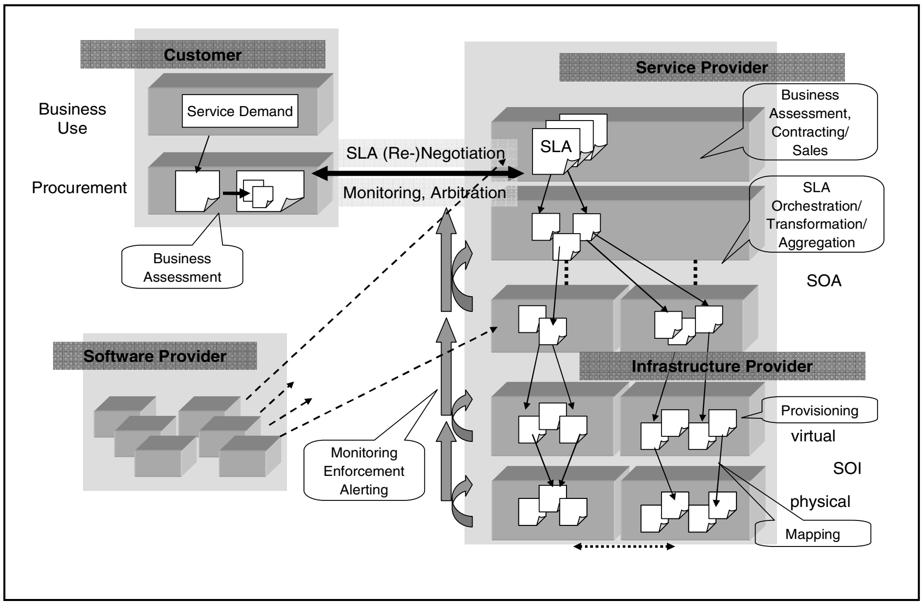


**Fig. 1.** Envisaged interaction of SLA stakeholders

Figure 1 gives a simplified overview of this systematic SLA management process. As today's business systems typically consist of complex layered systems, user-level SLAs cannot be directly mapped onto the physical infrastructure. Services might be composed of other more fundamental services that could be even provided by external parties. Consequently, a stepwise mapping of higher-level SLA requirements onto lower levels and the aggregation of lower-level capabilities to higher levels is crucial for grounding user-level SLAs to the infrastructure. This vertical information flow must carefully reflect service interdependencies as well as the originating business context. In addition to SLAs, the vertical information flow also covers monitoring, tracking, and accounting data and must support brokering and negotiation processes at each layer. As shown in the figure, the overall SLA management process may include different stakeholders, namely customers, service and infrastructure providers, and also various business steps such as business assessment, contracting and sales. The overview is intentionally simplified in the sense that no service chains are visualized. Such chains would represent all cases where service providers rely on additional external providers. Finally, the figure also shows the role of the software provider in charge of creating components with predictable behaviour.

The main challenge in this context is the integrated consideration of multiple facets as there are multiple stakeholders (software/service/infrastructure providers and customers), multiple roles (business / IT people, experts, users), multiple layers (on business and IT level), multiple service types (human-centric services, software services, …), various service level aspects (security, performance, …) and the consideration of the complete service lifecycle (engineering, composition, negotiation, provisioning, operation, monitoring, adjustment, decommissioning, …).

The integrated research into these aspects requires at least consideration of the following aspects:

1. The design and implementation of a core SLA management framework that relates perspectives of relevant stakeholders including:
   - Standardized models for SLA descriptions at different layers of a service-oriented architecture.
   - Concepts and algorithms for translating SLA descriptions between layers.
   - Methods and tools for multi-layer SLA management including planning, optimization, and provisioning.
   - Methods, tools, and techniques for monitoring (classic and predictive) and accounting services and SLAs.
2. The design and implementation foundations of an adaptive SLA-aware infrastructure including:
   - Standardized interfaces for adaptive infrastructures which allow for harmonized access to different virtualization technologies.
   - Advanced technologies for SLA enforcement and adjustment on infrastructure level by exploiting advanced virtualization technologies.
   - Advanced management technologies for service-oriented infrastructures by exploiting advanced virtualization technologies.
3. Advances in the engineering of predictable service-oriented systems by methodologies, modelling techniques, and prediction tools covering SOA and SOI components including
   - Prediction methods aware of usage profiles and supporting underspecified environments.
   - Manageability support by design.
4. To design and implement a comprehensive business management suite for e-contracting that covers the complete business lifecycle of a service provisioning/delivery including
   - Methodologies for end-user SLA negotiation and standards for end-user SLA specifications, heavily building upon recent proposed standards and research results (such as WS-Agreement and NextGrid).

The research project SLA@SOI addresses the integrated research on these challenges and will provide an open source SLA management framework that can be integrated into different service-oriented infrastructures.

## 3 Scientific Challenges and State of the Art

On the business level, there is a need to create a standardized business SLA model, common interfaces and tools to cover the whole commercial lifecycle addressing the whole business interaction between service providers and customers from the commercial product definition to post-sale relationship. One of the main challenges on the business level are the automatic merging of SLAs offered by different services into a final business SLA, something that does not exist today. Similarly, service-level objectives need to be translated towards the business objectives. Overall goal is the improved and well-determined quality of customer experience.

Service management plays a key enabler role for the holistic multi-layer SLA management challenge. Service management itself encompasses a variety of functional dimensions each of which tries to solve a different set of problems. Firstly, software components that make up the end services must be amenable to management related operations and expose introspection features. The second aspect is elaborate modelling of the service and IT infrastructure landscapes. The modelling practice includes expressive description of the entities within the landscapes, their functional attributes and operational interfaces, relationships and associations among the entities, constraints etc. These elaborate models can be applied to change impact analysis, what-if type scenarios etc.

SLA-aware service management has implications on service discovery and composition. The implications of SLAs on discovery are particularly manifested in contexts where dynamic composition of services is required. Dynamic composition usually implies dynamic discovery, selection and engagement of service implementations. Dynamic discovery and composition may further entail both translation of SLAs from different IT levels and consideration of the current consumer/provider execution contexts. SLA@SOI will give the opportunity to study these issues in depth and provide concrete solutions.

Concerning manageability, many standards have been proposed for implementing management interfaces [4]. Also, a variety of proprietary solutions for instrumenting components is available. So far, a development approach is missing that sufficiently supports a tight integration of manageability concerns for an SLA-driven management at design-time, while abstracting from specific technologies used for implementing the management interfaces and the instrumentation.

For such a heterogeneous and scaleable infrastructure layer suitable for SLA-aware service deployment, key requirements are appropriate abstraction, autonomic self-manageability, enforceability of contracted SLA's and alerting of violations, as well as the necessary accounting. Current state of the art is centered on policy-enabled autonomics, primarily fault detection and failover, and resilience to attack. Advanced diagnostics using statistics-based predictive models are emerging, applied to resource management and failure model analysis.

Virtualization interfaces are not harmonized to a level where they can be dynamically selected and interchanged [2]. Through SLA@SOI, introduction of SLA mapping with enforcement support will significantly benefit dynamic optimization, by reflecting criticality from upwards in the stack. Application of predictive analytics to resource consumption will enable proactive and pre-emptive reconfiguration decisions to safeguard SLAs and maximize efficiency. SLA@SOI will provide harmonized management interfaces for various virtualization/Grid-technologies thus enabling their transparent usage depending on SLA-driven requirements.

With regard to SLA management, there is a significant amount of ongoing research (e.g. [5]), but most of the relevant issues still remain open. The part relevant to modelling SLAs is being addressed to a large extent by the Open Grid Forum and the WS-Agreement specification [6]. WS-Agreement defines a representation of SLA templates with terms free to modify, SLA offers based on these templates, and agreements themselves. However, negotiation of SLAs is only starting to be discussed. Suitable negotiation strategies for different use cases are one of the core subjects that SLA@SOI will investigate. Developing a negotiation protocol that can fit all these strategies is also part of the work; such a protocol would have to deal with lack of reliable message delivery, clock synchronization issues, race conditions and change of landscape. In the same context, advance reservation and co-allocation of resources is a topic that, although significantly researched, there are no clear solutions. Additional complexity is inserted by the multi-layer nature of our architecture.

The optimization of SLA allocation, typically taking into account multiple criteria and objectives, is an active research topic with methods ranging from evolutionary algorithms to decision support systems. Even after the SLA is established, its monitoring and the reaction to exceptions is an area without pre-existing common solutions, where the project will have to provide them. SLA monitoring using events coming from the infrastructure and the services, their correlation and analysis in relevance to the SLA terms and the policies in place, as well as the action to take in the case of exceptions, are issues to consider and research. Re-provisioning, re-negotiating and fallback to penalties are the most evident options; populating any of the three approaches up the SLA hierarchy invokes dependencies that have not been explored yet.

As the provided services have to adhere to certain quality requirements at run-time, it should be possible to predict the quality of service on basis of design-time models. This allows evaluating different design alternatives with regard to the resulting quality and enables the service provider to estimate feasible service offerings depending on the particular environment. Therefore, an engineering methodology for developing predictable software services is required. Currently no model-based prediction approach [7] explicitly considers all relevant factors influencing the quality of software modules. Particularly the internal composition of components and the usage profile are supported insufficiently. Furthermore, the specifics of service-oriented application design and the flexibility offered by virtualized infrastructures as the software's deployment context are not sufficiently regarded.

## 4 Conceptual Architecture

The first challenge for designing an SLA management framework is to create a conceptual architecture which consistently bridges/mediates the various views of stakeholders and business/IT layers. We present this architecture in terms of *data view* and *functional view*. The functional view is further divided into *negotiation/provisioning-time* and *run-time*, respectively.

Firstly the data view of the conceptual architecture is shown in Figure 2. The active data sources in the figure are the monitoring events from the software and infrastructure components, while others are passive data stores. On one hand, the infrastructure and software landscape include the collective state of the provider(s) within the context under examination, for instance, the set of middleware/application packages and

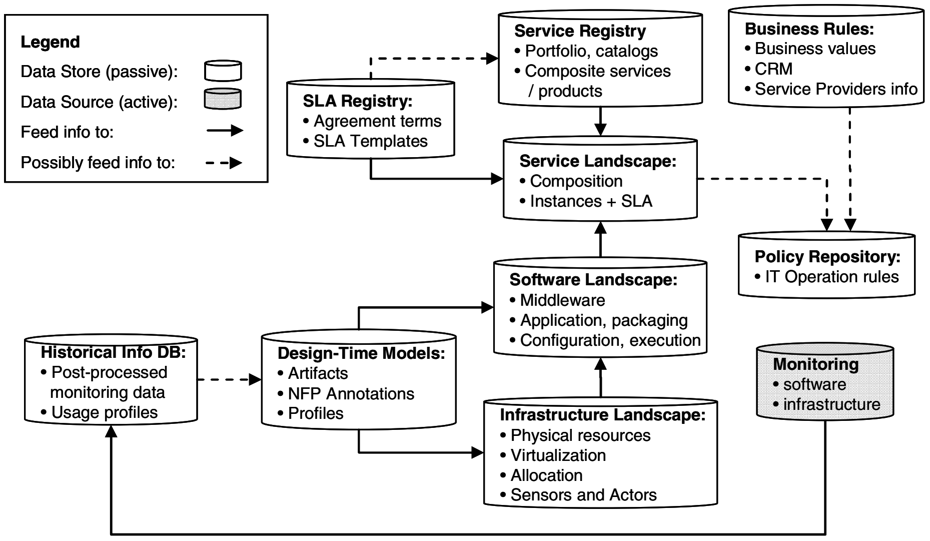


**Fig. 2.** The *data view* of the conceptual architecture: data stores and main direct relationships

their configurations in the software landscape. The design-time model (e.g. for capacity planning) may use data from the historical info DB, and it feeds information into both infrastructure and software landscape. On the other hand, the service landscape needs information from the software landscape, and it receives data from both service and SLA registry. For defining operation rules in the policy repository, information may be required from the business rules and the service landscape. Following such a high level data view a more detailed information flow analysis will be conducted in a later stage.

Secondly the negotiation and provisioning-time functional view of the architecture is sketched, as is shown in Figure 3. A typical control flow starts when a negotiation process is initiated by customer or provider. The core SLA management framework on the provide side, as is shown in the centre of the figure, consists of three main functional modules. They are SLA *negotiation*, *translation*, and *planning/optimization*. When an SLA is being negotiated, it needs to be translated across the whole IT stack in order to construct possible SLA hierarchies. These are then further planned/optimized in a cost-effective way under facilitation of the design time prediction.

Naturally the negotiation module requires higher level information from SLA/service registries and business rules. The translation and planning/optimization counterparts, on the other hand, needs lower level information concerning the software and infrastructure landscape. When negotiation is completed, it enters into the phase of SLA provisioning. By incorporating information from the current service landscape the provisioning module will invoke physical or virtual resource allocation and software deployment, which follow the operation rules in the policy repository. This view clearly shows a multi-level SLA management approach covers the whole scope from customer contracting down to infrastructure provisioning.

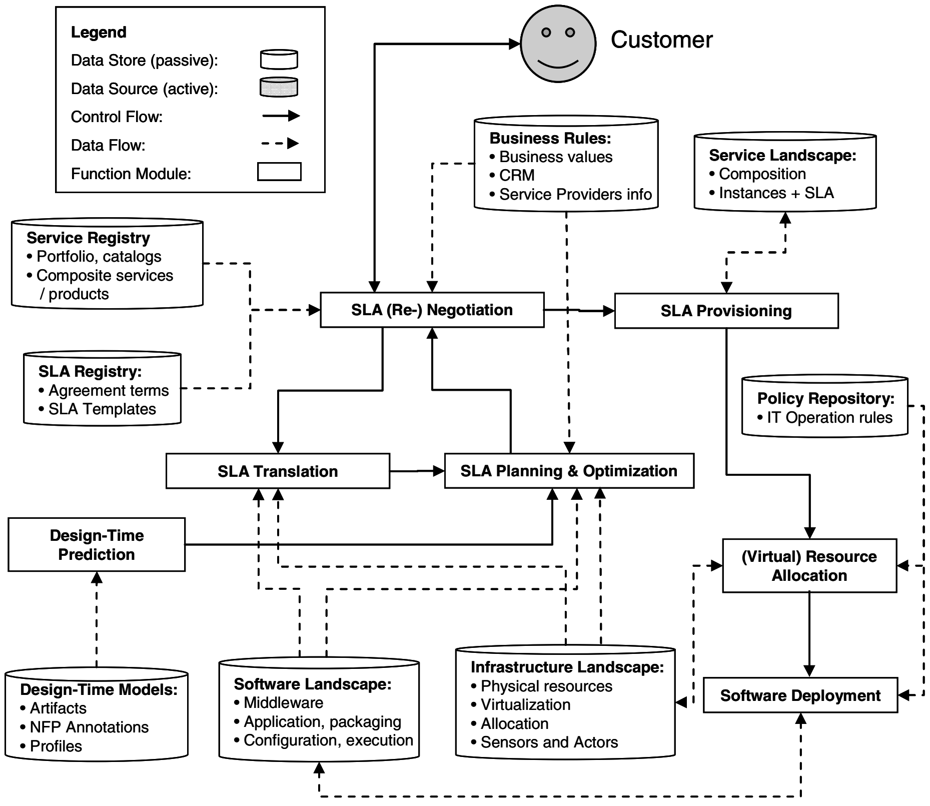


**Fig. 3.** The negotiation and provisioning-time functional view of the conceptual architecture

Thirdly the run-time functional view of the architecture is shown in Figure 4. At the bottom of the figure there are two capability modules, namely, monitoring and steering. In run time, standard monitoring data from the infrastructure and software are collected and made available to other components. This monitoring information can be processed in real-time for presenting SLA-specific metrics (SLA monitoring), for correlating and detecting events, and can be used for run time predictions. Such predictions, in turn, feed crucial information into the autonomic management modules which steer the underlying software and infrastructure for quality of service (QoS) adjustments. The SLA monitoring aggregates all kinds of monitoring data for detecting (possible) SLA violations. Simple SLA violations might be immediately resolved by an SLA adjustment. More complex ones trigger the SLA Planning & Optimization module for constructing solutions that span across components and layers. Eventual SLA adjustments are done via the standard steering capabilities. The SLA Interface Access Layer is the visual access point for humans to exercise influence on SLA management decisions. The rest of the functional modules and relationships are similar to the negotiation/provisioning-time view discussed above.

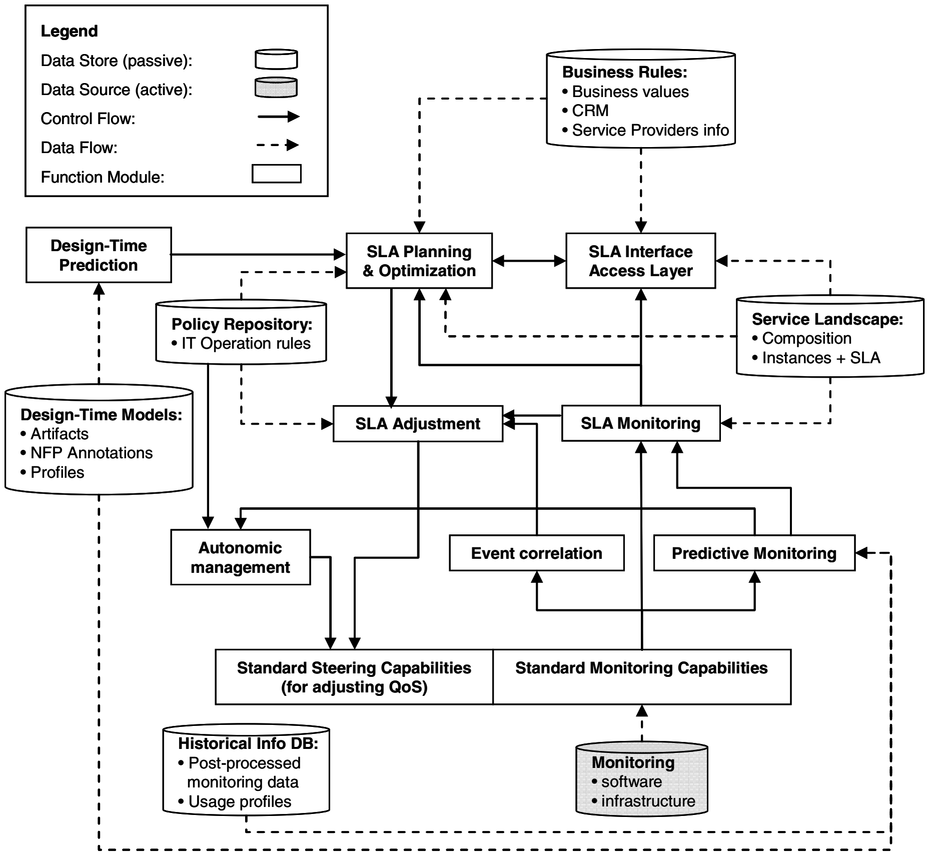


**Fig. 4.** The *run-time functional view* of the conceptual architecture

It should be noted that also the runtime view implies a hierarchical view on SLA management as all the various modules may exist at different levels and need proper synchronisation between them.

## 5 State of Practice in SLA Management

A range of industrial and government use cases have been selected to assess and demonstrate the impact of SLA mapping in the real world. Reflecting the current state of practice of service provisioning in various important contexts, in the real world, they are an important means of validating the results of SLA@SOI.

**Enterprise Resource Planning (ERP).** ERP software, as provided e.g. by SAP, belongs to the most complex existing software artefacts in the world. Thousands of software components with thousands of configuration and deployment options can be flexibly combined in order to satisfy highly specific customer needs. For managing this complexity, SAP defined a sophisticated solution lifecycle including a well-defined implementation phase for each customer setup where thorough consistency checks are done before any solution goes live. Technically, an SAP system comprises multiple architectural layers, including technology platform, business process platform, applications architectures, service architectures and system landscapes. The integrated management of non-functional/quality issues is largely done via humans based on guidelines, checklists and various loosely related support functions. SLAs are typically either negotiated manually per customer setting or on the basis of predefined system templates.

The main challenges for a more automated management of quality issues and SLAs are manifold [8]. Just as example, the main ones relevant for the area of efficiency are listed below: First, there is the underspecified environment which means that (a) concrete deployment and infrastructure are unknown at design time, (b) customer requirements/behaviour are unknown at design time and still underspecified at go-live time, (c) actual control flow is known vaguely at design time and slightly better at testing time (scenario-based) again better after business configuration and even better at run-time, (d) component developers are focussed on one architectural layer while non-functional characteristics of lower layers are only vaguely specified and subject to change and (e) the number of configuration & usage variants prevents from exhaustive testing. Second, the various architectures and programming models are just loosely coupled which means that no formal/provable relationship between architecture models and programming artefacts exists. It is currently unclear whether a closer coupling is feasible at all with general purpose programming languages. Third, technical expertise on non-functional behaviour of artefacts is widely spread and poorly formalized, so it's hard from an overall perspective to say who knows/does what and when.

**Enterprise IT implementation and operation.** In an enterprise context, strategic, operational (supply chain) and support functions, each make use of specific enterprise application and platform configurations which present varied workload and criticality patterns to the IT infrastructure. Currently, infrastructure allocation follows the traditional static / dedicated model with virtualisation technology mainly seen as a consolidation opportunity. The scope of SLAs is restricted to fulfilment of design-time functional requirements, and operational performance issues such as availability, response time and capacity. The mapping between high level SLAs, and configurations and policies of IT services and platforms, if it exists at all, is manual and defined in operational documents and procedures.

While this provides a reasonable safeguard to enterprise service delivery, it makes platform capacity allocation inefficient in terms of run time utilisation. Additionally it hinders the timely allocation of compute resources on the basis of business criticality and value of services.

Enterprise Architecture as a practise grounded in IT, is maturing to the point where a very sophisticated and fine-grained view can be formed of the processes and services that make up the extended enterprise: their interdependencies, their business value both operational and strategic, and their requirements and costs. Most of these processes rely in turn on IT services and infrastructure, therefore IT investment in providing these services, and ensuring their performance, should be commensurate with the criticality and value to the business.

Virtualisation presents an opportunity to more flexibly allocate infrastructure but in order to operate dynamically, resource allocation needs to be significantly automated. This in turn requires that the rules which govern allocation of low level resources are readily reflective of upper level SLAs, and mapping is end-to-end. The Enterprise use case will assess and demonstrate the results of SLA@SOI, in terms of enabling such a dynamic and fine grained provisioning model whereby compute resources are optimally allocated to point of value.

**Financial Services.** The Financial Services Use Case provides a generalised grid scenario with emphasis on dynamic Virtual Organisation creation, Service Level Agreements and Service Assurance, Workflow orchestration, service deployment and federated access to Data/Compute resources. The use case is based around the typical requirements of the Finance sector in particular applications including Implied Volatility and Risk Management (analysing the risk of a portfolio of stocks/bonds). Some of these requirements build on and extend results from the NextGRID project [4]. The key objectives of this Use Case will include:

- Definition of generalised grid SLAs based on the use case scenarios, with a focus on specific non-functional sector requirements.
- Implementation of the prototype demonstrator with support for Virtual Organisation, security, Service Assurance, Workflow orchestration and dynamic service deployment requirements.
- Devising SLAs to support handling sensitive end-user information.

The financial sector depends heavily on process and data intensive computations to deliver competitive advantage. Financial applications are particularly suited to grid-based experimentation and research. Many applications involve both process and data intensive computations. Finance applications demand a high availability of resources. Non-availability of resources means an absence in market trading which, in turn, can lead to missed opportunities. Security is of paramount importance. In addition, regulatory issues exist within institutions that place restrictions on the accessibility of information across their distributed enterprises.

**E-Government services to citizens and enterprises.** The particular context of this domain are services to citizens and enterprises where we focus on the adoption of service agreements as a driving tool for the implementation of the Social Service Sharing System, a service oriented framework which covers in an integrated way the provisioning and monitoring of social and health assistance services. From the governance point of view, there is a need for monitoring and analysis of costs, performances and quality of the system as a whole and of the services provided by the different organizations involved in this scenario (government bodies, hospitals, nursing homes, social service providers). At the same time, the system will provide an integrated platform to the citizen, with access to health care and social services according to needs and the ability to monitor progress.

The fundamental role of service agreements in e-Government service provisioning has been already recognized in the context of the so called G2G (government-to-government) service provisioning. Their adoption is, on the contrary, still very limited in scenarios such as the Social Service Sharing System, which require also G2B (government-to-business) and G2C (government-to-citizen) service provisioning. The particular challenge on SLAs is that these are not only based on market rules, but they are most often driven by "social" agreements between public bodies and citizens. As a consequence, the SLA negotiation (both between public bodies and citizens and between public bodies and private service providers) is different than in market-oriented domains. Another challenge of this domain is that it requires an integration of human based services (e.g., home care, medical assistance at home, transport services, and so on) with IT services; the underlying service oriented infrastructure is hence not only a technological infrastructure, but also a social and organizational one.

## 6 Conclusions

This paper discusses the need for multi-level SLA management approaches in order to fuel the next step towards a service-oriented economy. Rather than considering SLAs at a singular level we argue for a comprehensive SLA management approach that spans across multiple stakeholders and layers of a business/IT stack. We detail the main aspects of such an SLA framework, provide a conceptual architecture and some insights into industrial practice in various domains.

As next step we intend to provide a first prototype including a fully open source demonstrator which will give full insight into our approach. Furthermore the open source nature of our results will allow others to do comparative experiments and extended research.

## References

1. CoreGRID, TR-0096: Using SLA for resource management and scheduling - a survey (August 2007), `http://www.coregrid.net/mambo/images/stories/Technical-Reports/tr-0096.pdf`
2. Nessi-Grid: Grid Vision and Strategic Research Agenda. Deliverables D.1.1 & D.1.3 from NESSI-Grid project (IST-033636) (October 2006), `http://www.nessi-europe.com/Nessi/Portals/0/Nessi%20repository/EU%20Projects/NESSI-Grid/Deliverables/NESSI-Grid-SRA_v1.0.pdf`
3. SLA@SOI project (IST- 216556; Empowering the Service Economy with SLA-aware Infra-structures), `http://www.sla-at-soi.org`
4. NextGrid project (IST- 511563; next generation Grid architectures), `http://www.nextgrid.org`
5. Papazoglou, M.P., van den Heuvel, W.J.: Web services management: a survey. IEEE Internet Computing 9, 58–64 (2005)
6. The Open Grid Forum: Web Services Agreement Specification (March 2007), `http://www.ogf.org/documents/GFD.107.pdf`
7. Balsamo, S., Di Marco, A., Inverardi, P., Simeoni, M.: Model-Based Performance Prediction in Software Development: A Survey. IEEE Transactions on Software Engineering, 295–310 (2004)
8. Theilmann, W., Kilian-Kehr, R.: Quality Considerations in SAP Architectures. In: CompArch 2008, Industrial Experience Report Track (October 2008), `http://comparch2008.ipd.uka.de`

# Author Index