I-WEST 2008

José Cordeiro, Marten van Sinderen and
Boris Shishkov (Eds.)

# Enterprise Systems and Technology

Proceedings of the
2nd International Workshop on
Enterprise Systems and Technology - I-WEST 2008

Enschede, The Netherlands, May 2008

Organized by

IICREST

Co-organized by

CTIT

In cooperation with

INSTICC

INSTICC PRESS

José Cordeiro
Marten van Sinderen
and Boris Shishkov (Eds.)

# Enterprise Systems and Technology

**Proceedings of the**
**2nd International Workshop on**
**Enterprise Systems and Technology**
**I-WEST 2008**

Enschede, The Netherlands, May 2008

**INSTICC PRESS**
*Portugal*

Volume Editors

José Cordeiro (Portugal)
Marten van Sinderen (The Netherlands)
and Boris Shishkov (The Netherlands)

2nd International Workshop on
Enterprise Systems and Technology
Enschede, The Netherlands, May 2008
José Cordeiro, Marten van Sinderen and Boris Shishkov (Eds.)

Printed in Portugal

ISBN: 978-989-8111-50-0
Depósito Legal: 276371/08

# Foreword

This volume contains the proceedings of the Second International Workshop on Enterprise Systems and Technology (I-WEST 2008), held on May 23 in Enschede, The Netherlands.

The I-WEST workshop is a scientific event of *IICREST*, the *Interdisciplinary Institute for Collaboration and Research on Enterprise Systems and Technology*. I-WEST aims at providing a platform to researchers and practitioners, from academia and industry, to discuss challenges, solutions, ideas and experiences related to the broad field of enterprise systems and technology. Each year, a special theme is chosen within this broad field, in order to make presentations and discussions more focused. The theme of I-WEST 2008 is: ***Design of complex enterprise systems and aligned IT services***.

The past decade has witnessed important advances in both design approaches and technology-driven system capabilities. On the one hand, modeling approaches such as SOA and MDA have been complemented with techniques for expressing design intent at higher abstraction levels and keeping abstract models in sync with implementation-oriented models, thus increasing control over the design process. On the other hand, wireless networks, mobile computing, sensor networks and agent technologies created new application opportunities, primarily aiming at increasing availability, ease-of-use and sophistication of system capabilities, but at the same time leading to more complex systems with corresponding complex design processes.

Although current enterprise technology essentially concerns the widely considered challenge of overcoming the business-technology gap, there are emerging requirements to enterprise systems, which further complicate the enterprise engineering process. Such requirements relate to desirable qualities concerning enterprise systems, such as contex-awareness, pro-activity, intelligence. Hence, these new demands need to be explicitly addressed by application developers.

The goal of this workshop is to look at issues related to designing complex enterprise systems and aligned IT services. We are particularly interested in application opportunities offered by context-awareness, the design challenges that the incorporation of such properties in distributed environments bring, and how these challenges can be addressed in the design of enterprise systems that exploit these opportunities and IT services that provide appropriate support. I-WEST 2008 also has the intention to bring together researchers from various communities, including researchers working on enterprise engineering, context-aware systems, IT infrastructures, model-driven development and service-oriented architectures.

Following the I-WEST 2008 Call for Papers and received submissions, 6 papers were selected for a 30-minutes oral presentation during the workshop and for publication in these proceedings. The selected papers are a good illustration of different relevant topics that are currently under research: some papers are more oriented towards application development (considering this from the perspectives of enterprise ontology, requirements elicitation, and context awareness) while others are more SOA-oriented (considering service composition, utility computing, and process mediation).

Taking this opportunity, we would like to express our sincere gratitude to all people who have contributed to I-WEST 2008, including the authors (who have provided the main content for this workshop) as well as the program committee members and reviewers (who have provided constructive comments contributing to the quality of the content). We would also like to thank Vitor Pedrosa, for the brilliant organizational support. Finally, we tremendously appreciate the willingness of INSTICC to publish the proceedings, expressing respect and gratitude to its President, Joaquim Filipe.

We wish all presenters and attendees an inspiring workshop, and a nice stay in the beautiful city of Enschede.

May 2008

José Cordeiro
Marten van Sinderen
Boris Shishkov

## Workshop Chairs

## Program Committee

## Supporting Organizations and Projects

CTIT - Centre for Telematics and Information Technology

INSTICC - Institute for Systems and Technologies of Information, Control and Communication

Freeband A-MUSE project - Architectural Modeling Utility for Service Enabling in Freeband

# Table of Contents

## INVITED SPEAKERS

# Service Innovation: A Multi-Disciplinary Approach

Bart Nieuwenhuis

University of Twente, School of Management and Governance
Enschede, The Netherlands
l.j.m.nieuwenhuis@utwente.nl

**Abstract.** The market service share in Western European economies is growing at cost of agriculture and manufacturing. The success of these economies is more and more depending on the success of their service economy. The majority of the jobs, GDP and productivity growth depends on service innovation. The service sector accounts for more than two thirds of deployment and Gross Domestic Product (GDP) and Gross Value Added (GVA). During the last decades, the services sector is the only economic sector that has generated jobs. New, innovative services are the major source of economic growth in the years to come. The introduction of new services to the market is one of the major challenges for service companies in western economies.

Information and communication technology can be an enabler and a driver for service innovation. The penetration of the Internet and mobile phones are examples of these developments. These developments also illustrate the globalization of previously national service markets. Consequently, the scale at which services can be deployed is unprecedented.

However, service innovation is a complex process and certainly not only driven by technological advances alone. In general, service innovation is multi-dimensional and requires besides technological changes also new or adapted service concepts, new ways of interactions with customers and suppliers and new or changed processes within the organization of service providing companies. Research shows that innovation in the service company differs from innovations in a manufacturing company in various ways.

Companies are heading for a more systematic approach to develop new services, but have difficulties to find employees with the right mix of competences. Policy makers are developing innovation programs that stimulate service innovation, but have limited knowledge on service innovations. The academic institutes and research organizations have difficulties to conduct research programs due to their mono-disciplinary organization structure.

In this keynote lecture, we present the results of a collaborative project where service companies, research organizations and governmental organizations have developed a multi-disciplinary, multi-sector program to stimulate service innovations. We give an overview of the various dimensions that can be used to elaborate on services and service innovation. We also present a service innovation research agenda based on the results of interviews expressing the needs of more than thirty service companies in The Netherlands.

## Brief Biography

Bart Nieuwenhuis is part-time professor at the School of Management and Governance at the University of Twente. He is member of the Research Group Information Systems and Change Management (ICMS), holding the chair in QoS of Telematics Systems. He is working as advisor and consultant for his own consultancy firm K4B Innovation.

His research focuses on generic service provisioning platforms including Quality of Service mechanisms. Application domains comprise telemedicine as well as billing and payment services. His research interests include service innovation and business modelling. Bart Nieuwenhuis supervises PhD students and publishes scientific articles and conference papers on services provisioning platforms and middleware technologies for Quality of Service and Context Awareness. Bart Nieuwenhuis is chairman of the innovation-driven research programme Generic Communication, part of R&D programmes funded by the Ministry of Economic Affaires.

For K4B Innovation, Bart Nieuwenhuis works as an advisor to The Netherlands ICT Research and Innovation Authority. He was one of the initiators of EXSER, a centre of service innovation in The Netherlands. This centre is currently founded and is expected to start in the second half of 2008. The centre is sponsored by various large, innovative service companies and governmental organizations in The Netherlands.

Before joining the ISCM group, Bart Nieuwenhuis was part-time full professor at the Architecture and Services of Network Applications (ASNA) group within the Faculty of Electrical Engineering, Mathematics & Computer Science (EEMCS) of the University of Twente. He joined the ASNA group in Twente after a period of five years at the University of Groningen, where he was Tele-Informatics professor at the Computer Science Faculty.

Before starting his own company, he worked more than 20 years for KPN Research, the R&D facility of KPN, the telephony and Internet market leader in The Netherlands. He served as manager of R&D departments and Head of Strategy of KPN Research. Bart Nieuwenhuis worked on behalf of KPN for the European Institute for Research and Strategic Studies in Telecommunications (EURESCOM) in Heidelberg and was leader of various international, cooperative projects of European public network operators. Bart Nieuwenhuis holds a PhD in Computer Science and a MSc (cum laude) and BSc in Electrical Engineering, all from the University of Twente.

# PAPERS

# A UML Profile for Enterprise Ontology

José Cordeiro[1], Joaquim Filipe[1] and Kecheng Liu[2]

[1]EST Setúbal/IPS, Rua do Vale de Chaves, Estefanilha
2910-761 Setúbal, Portugal
j.cordeiro@computer.org, j.filipe@est.ips.pt
[2]The University of Reading, Whiteknights, Reading, RG6 6AF, U.K.
k.liu@reading.ac.uk

**Abstract.** Enterprise Ontology (EO) is a new subject that applies the Ψ-theory to the development and conception of social information systems. This theory, proposed by Jan Dietz, is based on the Language Action Perspective. In order to model an enterprise this theory presents a modelling method composed by four distinct aspect models, which use a collection of tables and diagrams to express themselves. A domain specific language was supplied for those diagrams that it is not standard or easily portable. In order to make these diagrams more useful and available this paper proposes the use of the Unified Modelling Language (UML) to represent EO most important diagrams where the EO main concepts are shown. The use of UML will bring some important benefits such as portability, interoperability, wider audience and understanding among others. In this sense a UML 2 profile has been created for representing the diagrams mentioned. An example of application of this profile is shown and an extended discussion of its creation is made. This will address the difficulties and issues found when metamodelling the solution using UML and will help to assess the feasibility of UML for this kind of problems.

## 1 Introduction

The Ψ-theory proposed by Jan Dietz [2] provides the foundations for designing and engineering of enterprises seen as social information systems. This theory captures the stable essence of any organization by focusing on commitments and the way people interact using language. This focus on language rather than material actions or technology comes from the Language Action Perspective view of the world (see, for example, [14]) in which it is based. The Ψ-theory comprises a modelling method composed by four distinct aspect models: the construction model, the process model, the action model and the state model. A methodology named "Design and Engineering Methodology for Organizations" (DEMO) is the basis for this modelling method [2], [3]. As many modelling methods, a few diagrams are used as a way of expression. Because these diagrams constitute the main communicational mean for representing the structure of the enterprise according to the Ψ-theory we are interested in include these diagrams together with other diagrams in information system development projects. This is not easy due to the proprietary language used by these diagrams and its lack of interoperability with other modelling languages. Also a formal verification is not automatically made and there are only a small number of tools to work with those diagrams. To overcome these problems we propose to use the Unified Modelling Language (UML), to express the most important diagrams of those models. UML is today a *de facto* standard widely used for modelling purposes, having numerous tools available and its use will bring us some important benefits such as:

- A wider audience will be able to use and understand the diagrams.
- Diagram interoperability and inclusion in software projects
- Formal representation and automatic verification of the diagrams

In this sense this paper presents and proposes a new UML 2 profile for representing those different types of diagrams. The difficulties and issues raised in the profile creation will also be the focus of this work because they will show the feasibility and the problems of using UML for such purposes.
This paper is organised as follows: section 2 presents related work, section 3 summarizes the main concepts of Enterprise Ontology (EO), the proposed UML Enterprise Ontology profile will be shown and exemplified in section 4, section 5 will present issues and rationale related to the EO profile development and finally, conclusions will be given in section 6.

## 2 Related Work

We found just a small number of papers that refer to the use of UML with DEMO. In [13], Shishkov and Dietz suggest using DEMO to derive UML use cases. In this work a mapping from DEMO Business Transactions to UML Use Cases is proposed. In fact there is a general tendency to separate the use of DEMO and UML. For example, in [7] DEMO is used to model the business processes prior to information systems modelling using UML. Nevertheless, the most significant work relating UML and DEMO is [11]. In this paper instead of a direct UML representation it is proposed a language mapping between DEMO models and UML. This mapping is accomplished in three phases: first a concept mapping between both languages is made, next a notational mapping is performed and at last there is a diagram transformation. In fact, a similar approach is taken when we create the UML profiles as we will see. In UML profiles concept matching is used to find the appropriate metaclasses corresponding to the DEMO model elements, also a notational option is taken for the created stereotypes and finally some new diagrams are created for showing the new model elements. Even so according to this paper it will be necessary to have the original DEMO diagrams instead of a direct UML representation as we pretend.
Regarding UML profiles much work has been done and some references will be pointed afterwards when appropriate.

## 3 Enterprise Ontology

Enterprise Ontology (EO) captures the essential aspects of any organization by focusing on the ontological level of business where people interact, commit themselves and

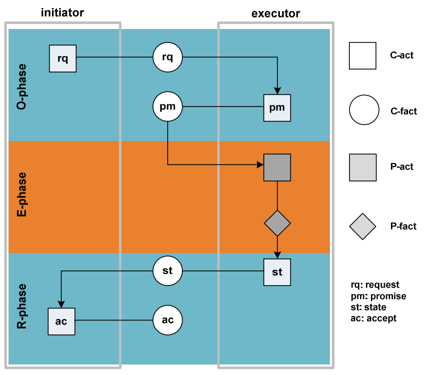


**Fig. 1.** The Basic Transaction Pattern (adapted from [3]).

produce results. At this level people use language acts as the driver of any business transaction or coordination acts. EO is about the construction and operation of an organization. The Ψ-theory establishes the basis and the theoretical support for EO. The Ψ-theory is composed by four axioms and one theorem. The first axiom – the Operation Axiom – presents an organization as a group of actors performing two kinds of acts: coordination acts (C-acts) and production acts (P-acts). C-acts are language acts used by actors to engage themselves in commitments and to ultimately originate the P-acts responsible for producing the effective work. The result of performing a C-act is a coordination fact (C-fact), whereas the result of performing a P-act is a production fact (P-fact) or production result. The second axiom – the Transaction Axiom – comes from the observation that P-acts and C-acts seems to occur in a universal pattern called *transaction*. This *transaction* is a key concept of the Ψ-theory and EO. The complete transaction pattern is seen as a socionomic law that underlies the conducting of any business always and everywhere. This *transaction* has its roots in the notion of conversation for action [14] and the Workflow Loop [8] both from the Language Action Perspective. In fig. 1 we depict the basic transaction pattern which has three phases: an order phase where the negotiation about the P-act to be executed takes place. In this phase two types of C-acts are usually performed: a request by the initiator actor and a promise to accomplish it by the executor actor. The next phase is the execution phase where the P-act is actually performed. Finally, the result phase ends the transaction with the performance of a C-act stating the completion of execution of the P-act by the executor actor and a C-act by the initiator actor accepting the result. The third axiom – the Composition Axiom – is concerned with the interrelation of P-facts in a production world (the P-world). In particular the enclosing relationship between transactions is analysed. Finally, the fourth axiom – the Distinction Axiom – is about the human abilities that have a significant role in performing C-acts namely the *performa*, *informa* and *forma* abilities. The *performa* ability is considered the *essential* human ability for doing business and is part of the ontological level of EO. The *Organization Theorem* completes the Ψ-theory by stating that “the organization of an enterprise is a heterogeneous system that is constituted as the layered integration of three homogeneous systems: the B-organization (from business), the I-organization (from intellect), and the D-organization (from Document)” [2, p.115].

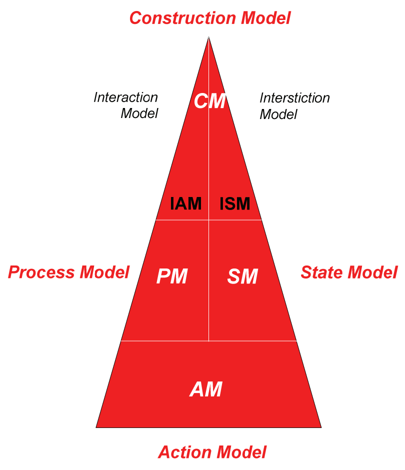


**Fig. 2.** The four DEMO aspect models (adapted from [2]).

For designing and engineering organizations EO is supported by the DEMO methodology. The DEMO methodology defines the required steps for that purpose and uses a modelling method composed by four distinct aspect models: the construction model, the process model, the action model and the state model that together constitutes the complete ontological knowledge of an organization (fig. 2). The construction model (CM) specifies transactions types, associated actors roles and information banks (conceptual stores of C-facts or P-facts). The CM is divided in two similar models: the *interaction model* (IAM) and the *interstiction model* (ISM) that shows us respectively the active and the passive influences between actor roles. The *process model* (PM) details the CM by showing the specific transaction patterns for each transaction type in the CM. The *action model* (AM) is the most detailed level and it specifies the action rules that serve as guidelines for the actors. The last model, the *state model* (SM) specifies the *state space* of the P-world. It includes object classes, fact types, result types and ontological coexistence rules. In general these models are expressed by different diagrams and tables. Table 1 show us the different diagrams and tables used by each of them and what they depict. As it is shown the AM doesn't use any diagram and the SM uses a very specific type of diagram that doesn't presents directly the main concepts of EO, namely the transaction types, and we decided not to represent them using UML. Thus, in this work we will be interested in provide UML diagrams to mirror the following diagrams: ATD, PSD and ABD.

## 4 The Enterprise Ontology Profile

In figure 3 a part of the metamodel for the UML Profile created for EO is presented. In this metamodel it is shown the equivalent UML elements for the DEMO Actor Transaction Diagram (ATD) elements. The corresponding stereotypes and constraints for this profile are detailed in table 2. Discussion of the creation of the complete profile is made in the next section.

**Table 1.** DEMO aspect models.

| Model | Expressed by | Typical contents |
|---|---|---|
| Interaction | Actor Transaction Diagram (ATD) | Actor roles, transaction types and their connecting links |
| | Transactions Result Table (TRT) | Transaction and result types |
| Process | Process Structure Diagram (PSD) | C-act/C-result, P-act/P-result, causal and conditional links and responsibility areas of actor roles |
| | Information Use Table (IUT) | Process steps and object class, fact types or result types |
| Action | Action Rule Specifications (ARS) | Action rules |
| State | Object Fact Diagram (OFD) | Object classes, fact types, result types and existential laws |
| | Object Property List (OPL) | Property types, object classes, scales |
| Interstriction | Actor Bank Diagram (ABD) | Information banks, actor roles and information links |
| | Bank Contents Table (BCT) | Object classes, fact types, result types and production banks |

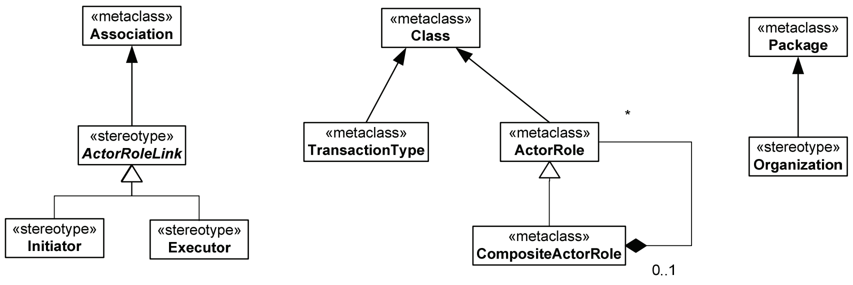


**Fig. 3.** EO profile metamodel part 1 - UML representation of ATD elements.

**Table 2.** EO stereotype definitions part 1 – ATD elements.

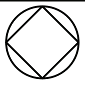
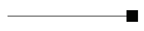
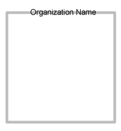
| **Name** | **TransactionType** | |
|---|---|---|
| **Extended Class** | **Class** | **Notation** |
| **Description** | Represents the transaction type concept. |  |
| **Constraints** | ------ | |
| **Notes** | The name of the transaction should be a capital T followed by the transaction number (ex: T02 ) | |
| **Name** | **Actor Role** | |
| **Extended Class** | **Class** | |
| **Description** | Represents the elementary actor role concept. | |
| **Constraints** | ------ | |
| **Notes** | No special notation. Usually it is shown as a rectangle with the actor role name inside. The actor role name should be a capital A followed by the actor role number (ex. A02) | |
| **Name** | **CompositeActorRole** | |
| **Base Class** | **ActorRole** | **Notation** |
| **Description** | Represents the composite actor role concept**.** | |
| **Constraints** | ------ | |
| **Notes** | The actor role name should be the capitals CA followed by the actor role number (ex. CA03) | As a class but filled using a gray colour |

**Table 2.** EO stereotype definitions part 1 – ATD elements(cont).

| **Name** | **ActorRoleLink** | |
|---|---|---|
| **Extended Class** | **Association** | |
| **Description** | A relationship between an actor role and a transaction type | |
| **Constraints** | 1) It is a binary association<br>2) Must connect a TransactionType and an ActorRole element<br>3) It is an abstract metaclasse | |
| **Name** | **InitiatorLink** | |
| **Base Class** | **Actor RoleLink** | |
| **Description** | A special kind of an ActorRoleLink that connects an ActorRole and a TransactionType where the ActorRole plays the role of the initiator of the transaction | |
| **Constraints** | ------ | |
| **Notes** | Usually no adornments are shown.<br>There is an implicit navigation from the actor role to the transaction. | |
| **Name** | **ExecutorLink** | |
| **Base Class** | **Actor RoleLink** | **Notation** |
| **Description** | A special kind of an ActorRoleLink that connects an ActorRole and a TransactionType where the ActorRole plays the role of the executor of the transaction |  The line end with the black square must be connected to the actor role |
| **Constrains** | ------ | |
| **Notes** | There is an implicit navigation from the transaction to the actor role. | |
| **Name** | **Organization** | |
| **Extended Class** | **Package** | **Notation** |
| **Description** | Represents a group of actor roles and transaction types which belong and take place inside an organization |  |
| **Constrains** | ------ | |
| **Notes** | The name of the package is placed upon its upper boundary line | |

A special UML diagram – the AT diagram – one of diagrams which we propose for this profile is a special case of a UML Class diagram. This diagram is used for showing actor roles and transaction types and the links between them and mimics the original DEMO ATD. In figure 4 an example of an ATD is given and in figure 5 the same diagram is reproduced with the EO profile applied to it.

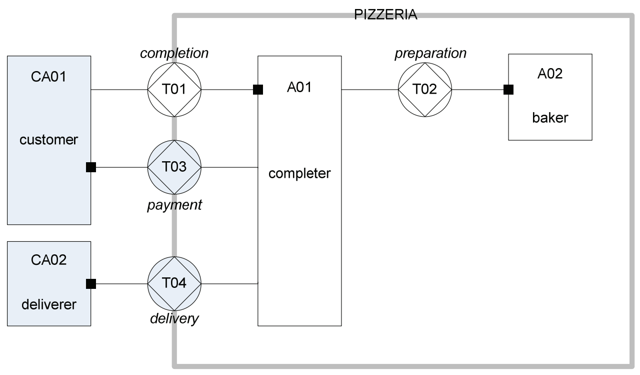


**Fig. 4.** Example of the DEMO ATD of a pizzeria (adapted from [2]).

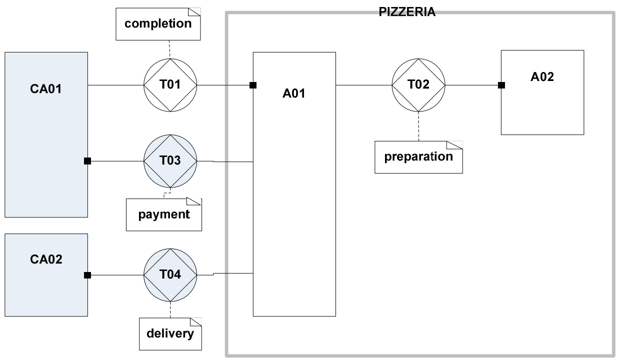


**Fig. 5.** An UML AT Diagram applied to the pizzeria example using the EO Profile.

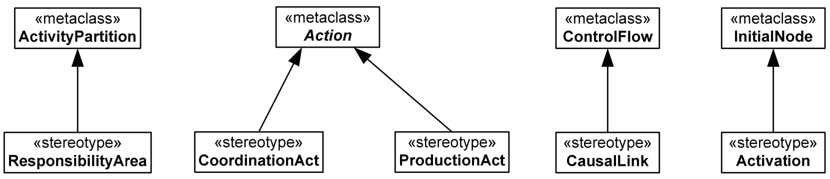

«metaclass» ActivityPartition
«metaclass» Action
«metaclass» ControlFlow
«metaclass» InitialNode
«stereotype» ResponsibilityArea
«stereotype» CoordinationAct
«stereotype» ProductionAct
«stereotype» CausalLink
«stereotype» Activation

**Fig. 6.** EO profile metamodel part 2 - UML representation of PSD elements.

In figure 6 it is shown the second part of the EO profile which includes the equivalent UML elements for the DEMO Process Structure Diagram (PSD) elements. The corresponding stereotypes and constraints for this profile are detailed in table 3.

**Table 3.** EO Profile stereotype definitions part 2 – PSD elements.

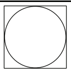
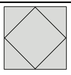
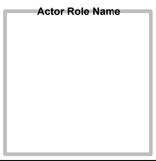
| Name | CoordinationAct | |
|---|---|---|
| Extended Class | Action | Notation |
| Description | Represents the C-act concept. |  |
| Constraints | ------ | |
| Notes | Just one symbol meaning the combination of a C-act with a C-result | |
| **Name** | **ProductionAct** | |
| Extended Class | Action | Notation |
| Description | Represents the P-act concept. |  |
| Constraints | ------ | |
| Notes | Just one symbol meaning the combination of a P-act with a P-result | |
| **Name** | **ResponsibilityArea** | |
| Extended Class | ActivityPartition | Notation |
| Description | Represents the responsibility area concept |  |
| Constraints | isDimension = true | |
| Notes | It will not be possible to have nesting of actor roles because each actor role establishes a unique dimension. | |

**Table 3.** EO Profile stereotype definitions part 2 – PSD elements(cont).

| | |
|---|---|
| **Name** | **CausalLink** |
| **Extended Class** | **ControlFlow** |
| **Description** | A link used to show the control flow between C-act and P-acts. |
| **Constraints** | ------ |
| **Notes** | It is equivalent of the causal link |
| **Name** | **Activation** |
| **Extended Class** | **InitialNode** |
| **Description** | It represents the start of a process. It can be placed inside a Responsibility Area meaning self activation or outside meaning external activation. |
| **Constraints** | ------ |
| **Notes** | |

As in the case of the AT Diagram a new UML diagram – the Process Structure Diagram - was created to show DEMO PSD using UML. This diagram is a special case of a UML Activity Diagram. Unfortunately UML diagrams are not part of the main specification of UML, they are not model elements, therefore we have to introduce these diagrams informally and it will not be possible to formalize some aspects of the diagrams, for example positioning rules for the included elements. An example of a PSD is given in figure 7. In figure 8 it is shown a UML PS diagram with the EO Profile applied.

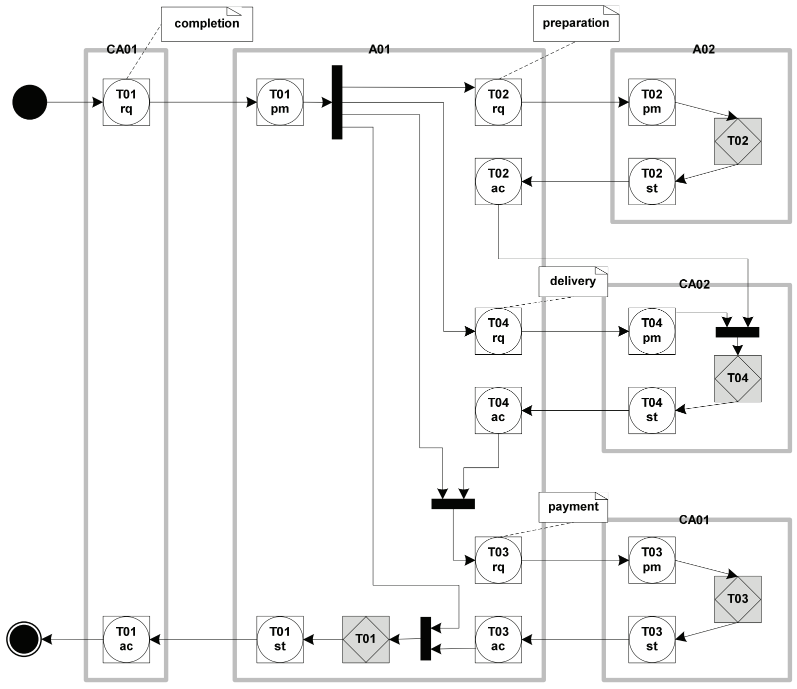


**Fig. 7.** An UML PS Diagram applied to the pizzeria example using the EO Profile.

Figure 9 shows the third and last part of the EO profile which includes the equivalent UML elements for the DEMO Actor Bank Diagram (ABD) elements. The corresponding stereotypes and constraints for this profile are detailed in table 4. Also a new

UML diagram – the Actor Bank Diagram - is proposed for representing the DEMO ABD. This diagram is also a special case of a UML Class Diagram. This diagram is very similar to the ATD.

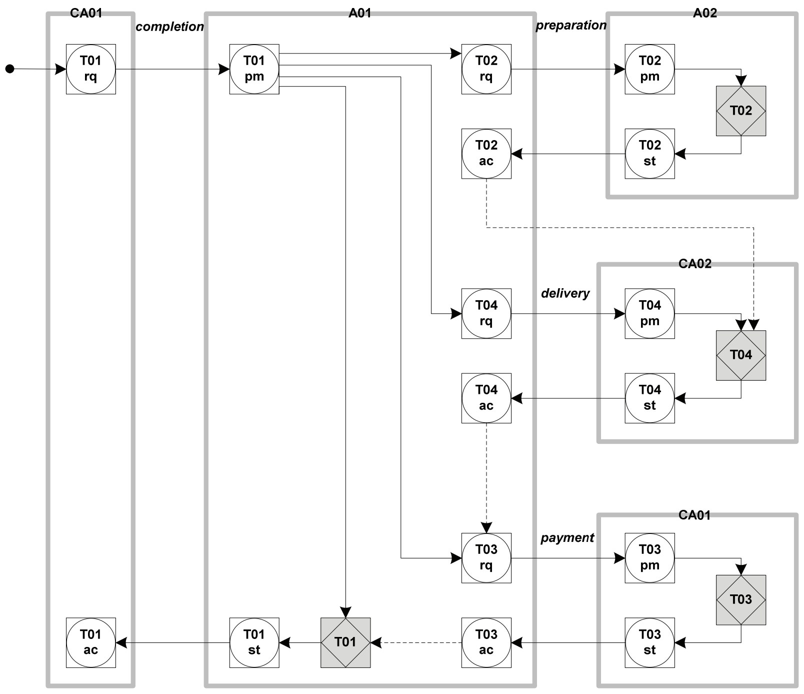


**Fig. 8.** Example of the DEMO PSD of a pizzeria (adapted from [2]).

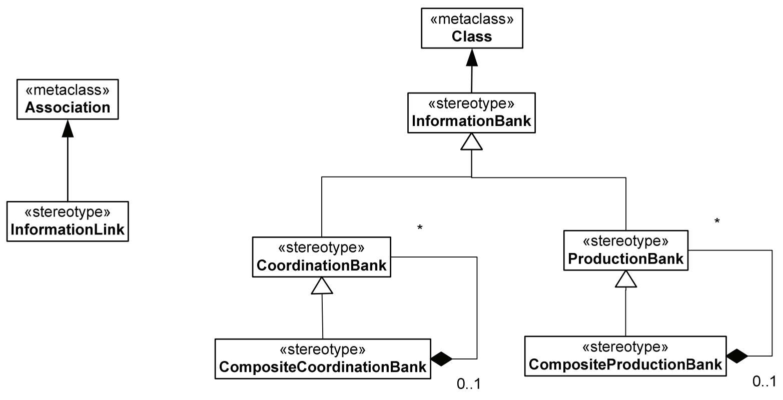

«metaclass» Class
«stereotype» InformationBank
«metaclass» Association
«stereotype» InformationLink
«stereotype» CoordinationBank
«stereotype» ProductionBank
«stereotype» CompositeCoordinationBank
«stereotype» CompositeProductionBank
*
*
0..1
0..1

**Fig. 9.** EO profile metamodel part 3 - UML representation of ABD elements.

**Table 4.** EO Profile stereotype definitions part 3 – ABD elements.

| | | |
|---|---|---|
| **Name** | **InformationBank** | |
| **Extended Class** | **Class** | |
| **Description** | A production or a coordination bank. | |
| **Constraints** | ------ | |
| **Notes** | | |
| **Name** | **CoordinationBank** | |
| **Base Class** | **InformationBank** | **Notation** |
| **Description** | Represents a coordination bank | ○ |
| **Constraints** | ------ | |
| **Notes** | The coordination bank name should be the capitals CB followed by the bank number (ex. CB02) | |
| **Name** | **CompositeCoordinationBank** | |
| **Base Class** | **CoordinationBank** | **Notation** |
| **Description** | Represents a composite coordination bank | ● |
| **Constraints** | ------ | |
| **Notes** | The composite coordination bank name should be the capitals CCB followed by the bank number (ex. CCB02) | |
| **Name** | **ProductionBank** | |
| **Base Class** | **InformationBank** | **Notation** |
| **Description** | Represents a production bank | ◇ |
| **Constraints** | ------ | |
| **Notes** | The production bank name should be the capitals PB followed by the bank number (ex. PB03) | |
| **Name** | **CompositeProductionBank** | |
| **Base Class** | **ProductionBank** | **Notation** |
| **Description** | Represents a composite production bank | ◆ |
| **Constraints** | ------ | |
| **Notes** | The composite production bank name should be the capitals CPB followed by the bank number (ex. CPB04) | |
| **Name** | **InformationLink** | |
| **Extended Class** | **Association** | **Notation** |
| **Description** | A connection relating actor roles and information banks | -------------------- |
| **Constraints** | 1) It is a binary association<br>2) Must connect an InformationBank and an ActorRole | |
| **Notes** | | |

## 5 EO Profile Creation

If we want to have the benefits of using UML tools such as model interchange, model validation and model storage it is necessary to create UML Profiles instead of a complete metamodel for representing the DEMO diagrams. Thus, and in order to build these profiles we should follow some guidelines (see for example [4]). In a general and simple view these guidelines recommend to create first a domain metamodel, to choose from this metamodel the relevant elements, to extend the appropriate UML metamodel elements with some of those elements and to define additional constraints and tagged values (see [12]). In our case we found as not necessary to create the domain metamodel because we already have the domain elements which correspond to the DEMO diagram elements. In spite of skipping this step and although simple the remaining process it has many issues, difficulties and compromises specially because we are metamodeling non object-oriented theories. In the next sections some of the issues and problems found for each DEMO diagram will be reported. It should be also noted that the EO UML profile that was created uses version 2.1.2 of the UML superstructure and infrastructure specifications [9], [10].

### 5.1 Development Issues - AT Diagram

The ATD diagram shows mainly actor roles, transaction types and links connecting them. For transaction types we don't have any similar UML model element and in this situation it is usual to extend the metaclass 'class' as a representation of a concept. Thus, the initiator or executor links between transaction types and actor roles should be expressed using the association metaclass. This is the most powerful relationship between model elements that is used to connect classifiers. The problem is the representation of actor roles. In UML we have an actor element that matches the concept of an actor role, it is a classifier and support associations as well but this element has limited capabilities compared to classes. So, the best solution was to express actor roles using classes as well. This will allow a more powerful expression of actor roles, it will permit to create composite actor roles based on elementary actor roles using the inheritance and composition concepts of object-orientation and it will have some extra benefits when creating the EO profile elements for the PSD DEMO diagram. The last ATD element that we need to represent is the boundary of an organization. This is a grouping element that joins transactions, actor roles and links belonging to an organization, but some transactions with external actor roles are not placed completely inside the organization boundary but are seen as belonging to the boundary itself. This cannot be represented used the common UML grouping element, the package. Elements of a package belong to the package and not to its boundary. Other UML grouping elements such as the activity partition or the subject (of a use case) are not suited for this purpose, they can only surround very specific UML model elements and the related concepts don't match with our goals. So, we adopted as a solution to use the package extension but to limit the elements of the organization package for being transaction types where all the actors are organizational actor roles and all the actor roles belong to the organization. A possibility is to show the boundary transaction types using a second package which includes only boundary transactions.

Regarding the notation we choose to make the UML Profile elements close to the original notation used in the DEMO diagrams. UML allows some flexibility in the notation and this possibility to make it close to the original or to the traditional UML as identified in [1] is used by our solution.

### 5.2 Development Issues - PS Diagram

The DEMO PSD shows two combined symbols for correspondingly C-acts/C-results and P-acts/P-results. The links between these symbols are made using causal and conditional links. Also external and self-activation lines are used to represent the start of the depicted processes. A last element in these diagrams is a grouping element defining the responsibility area of an actor role. Giving the "business process" nature of these kinds of diagrams it would be most useful to represent them using an UML activity like diagram. For this purpose their elements should be equivalent to the typical UML activity elements. In fact this is possible because the main elements, C-acts/C-results and P-acts/P-results, can be represented by extending the UML action element. If we consider just the C-acts and P-acts, they are both some kind of action and they are suited to be represented as actions. We should make implicit the produced C-results and P-results; it will be possible to see them as the output of the corresponding actions. Regarding the notation we will make it close to the original PSD elements. We choose to represent C-acts and P-acts using the combined symbol and thus making explicit the associated C-results and P-results. The C-act/C-result symbol is depicted as a single UML element hiding the combined nature of the symbol. This symbol results from joining a circle – a C-fact (or C-result) type - and a square – a C-act type - but this combination cannot be made using UML. There is no possibility to combine two different UML elements without creating a new element with no connection to the original symbols. The same applies to the P-act/P-result symbol which uses a square for the P-act and a diamond for the P-fact (or P-result) type that it is represented as a single symbol. UML doesn't allow us to combine model elements but we took advantage of the flexibility in the notation. Regarding the causal link it is in fact a kind of a UML control flow causing the flow to jump from one act to the next. In the case of the conditional link there is no equivalent UML element but we can use some of the UML elements to produce the same effect as the conditional link. In the case of the conditional link appear at the end of a conditional branch it can be replaced by a decision node at the beginning of the branch and a merge node at the end. In case of appearing at the end of a concurrent branch it can be replaced by a fork at the beginning and a join at the end. This last case is illustrated in fig. 8. In the PSD also responsibility areas are used to delimit a group of acts performed by an actor role. In this case an extension of activity partition is suited for this goal and can play the same role. This solution is optimal because we choose before to use UML extended class elements for actor roles. Thus, actor roles will be the responsible agents for the corresponding C-acts or P-acts. At last for activation lines UML also provide model elements, we can use the *Initial node* of the activity diagram connected to a C-act using a control flow to express the starting point of a process. If this Initial node lies inside a responsibility area where it is connected to a C-act it means a self activation, otherwise if it lies outside the responsibility area it means an external activation. Just a final remark to point the necessity of including a final flow node to indicate the end of a process. There is no similar model element in the original PSD.

## 5.3 Development Issues - AB Diagram

The last part of the UML profile concerns the creation of the corresponding UML elements for the DEMO ABD. This diagram is similar to the PSD and it can use most of the elements defined before. We just need to express as well elementary and composite production and coordination banks and information links. These banks are just a kind of databases and can be shown using an extension of the UML class. Also we can use the object oriented mechanisms to differentiate from elementary and composite production and coordination banks. The ABD uses also a combined symbol for a production and coordination bank that refers to an information bank. An information as a general kind of bank can be expressed as a base class and we can use inheritance to derive the production and coordination banks. We lose the combined nature but we gain in expressivity. Finally the information link is naturally expressed as an extension of an association because it relates stereotypes of classes.

**Table 5.** Summary of identified UML issues.

| **UML issue** | **Comments** |
|---|---|
| A diagram is not an UML metamodel element | It is not possible to adequately formalize relationships between diagrams and model elements because the diagrams are not UML elements |
| UML metamodel grouping elements with limited options | The most important grouping UML element - the package – provides a special kind of grouping that doesn't allow representing elements that belong to two packages simultaneously; this is the case of boundary elements such as some of the transaction types in AT diagrams. Other UML grouping elements such as the Activity Partition and the Subject have limited application given the restrictions in the elements they may contain. |
| UML metamodel elements usually have hidden aspects | Some simple UML elements cannot be used to represent similar concepts because they cannot be freely associated with other elements. This is hidden and it is a consequence of the rigidity of the UML metamodel when defining these elements. In the case of the *actor* element its limited capabilities make it preferable to use classes to represent actor roles although an *actor* was a better matching concept. |
| UML metamodel elements without combinations among them | It is not possible to combine different UML elements in one joint element preserving the original meaning of the individual elements. The example was the C-act/C-result and the production bank/coordination bank elements which had unique symbols for the combined element. |

## 6 Conclusions and Future Work

In this paper a UML profile for Enterprise Ontology was introduced. This profile brings important benefits for the underlying DEMO methodology such as:

- Possibility to communicate the diagrams to information system and software development teams and to include them with other diagrams in the same projects
- Interoperability of the diagrams with other model tools
- Consistency, verifiability and formalization of the diagrams

Concerning the profile creation this paper has raised some issues that are resumed in table 5. It should be noted as well that the stereotypes created in this profile introduced a reduced number of constraints in order to have enough freedom when using UML. Some additional constraints can be added if there is the necessity of a more formal and rigid expression of the produced diagrams.

This work is part of a research project that has as a goal to create a unified and fundamental theory for software development that integrates some relevant concepts of three different socio-technical theories, namely Organizational Semiotics [6], the Theory of Organized Activity [5] and the Language Action Perspective represented in this paper by the Ψ-theory and the DEMO methodology. Concerning the UML profile development issues raised in this paper, they complement another group of issues identified in [1]. As a future work a UML profile of the new theory will be proposed that will share some of the elements and concepts presented in this paper and in [1].

## References

1. Cordeiro, J., Liu, K.: UML 2 Profiles for Ontology Charts and Diplans - Issues on Metamodelling. Proc. of EMISA 2007: 191-204, (2007).
2. Dietz, J. L.: Enterprise Ontology – Understanding the Essence of Organizational Operation. *In: Enterprise Information Systems VII*. Eds. C. Chen, J. Filipe, I. Seruca, and J. Cordeiro. Springer, Dordrecht, The Netherlands (2006)
3. Dietz, J. L.: The deep structure of business processes. Communications of the ACM 49, 5, 58-64. (May 2006)
4. Fuentes, L. and Vallecillo, A.: An Introduction to UML Profiles. UPGRADE, The European Journal for the Informatics Professional, 5(2):5-13 (2004). ISSN: 1684-5285.
5. Holt, A.: Organized Activity and Its Support by Computer, Kluwer Academic Publishers, Dordrecht, The Netherlands (1997).
6. Liu, K.: Semiotics in Information Systems Engineering, Cambridge University Press, Cambridge, UK (2000).
7. Mallens, P., Dietz, J., and Hommes, B-J.: The Value of Business Process modelling with DEMO Prior to Information Systems modelling with UML. In Proc. EMMSAD'01, Interlaken, Switzerland (2001).
8. Medina-Mora, R., Winograd, T., Flores, R., Flores, F. The Action Workflow approach to workflow management technology. In J. Turner and R. Kraut, Eds., Proceedings of the 4th Conference on Computer Supported Cooperative Work. ACM, New York (1992).
9. OMG: Unified Modeling Language Superstructure Specificacion, v2.1.2. Available: http://www.omg.org/spec/UML/2.1.2/Infrastructure/PDF/ (Apr 2008)
10. OMG: Unified Modeling Language Infrastructure Specificacion, v2.1.2. Available: http://www.omg.org/spec/UML/2.1.2/Superstructure/PDF/ (Apr 2008)
11. Rittgen, P.: A language-mapping approach to action-oriented development of information systems. Eur. J. Inf. Syst. 15, 1, 70-81. (Feb. 2006)
12. Rumbaugh, J., Jacobson, I. and Booch, G.: *The Unified Modeling Language Reference Manual (2nd edition)*, Addison-Wesley, Reading, MA (2005)
13. Shishkov, B. and Dietz, J.: Deriving Use cases from Business processes, the Advantages of DEMO. *In: Enterprise Information Systems V*. Eds. O. Camp, J.B.L. Filipe, S. Hammoudi, and M. Piattini. Kluwer Academic Publishers, Dordrecht/Boston/London (2004)
14. Winograd, T. and Flores, F.: Understanding Computers and Cognition. Ablex Publishing Corporation, Norwood, NJ, USA (1986).

# On the Design of Context-Aware Applications

Boris Shishkov and Marten van Sinderen

University of Twente, Department of Computer Science, Enschede, The Netherlands
{b.b.shishkov, m.j.vansinderen}@ewi.utwente.nl

**Abstract.** Ignoring the dynamic context of users may lead to suboptimal applications. Hence, context-aware applications have emerged, that are aware of the end-user context situation (for example, “user is at home”, “user is travelling”), and provide the desirable services corresponding to the situation at hand. Developing context-aware applications is not a trivial task nevertheless and the following related challenges have been identified: (i) Properly deciding what physical context to ‘sense’ and what high-level context information to pass to an application, and bridging the gap between raw context data and high-level context information; (ii) Deciding which end-user context situations to consider and which to ignore; (iii) Modeling context-aware application behavior including ‘switching’ between alternative application behaviors. In this paper, we have furthered related work on context-aware application design, by explicitly discussing each of the mentioned interrelated challenges and proposing corresponding solution directions, supported by small-scale illustrative examples. It is expected that this contribution would usefully support the current efforts to improve context-aware application development.

**Keywords.** Application development; Context-Awareness; Behavior modeling.

## 1 Introduction

Traditional application development methods do not consider the context of individual users of the application under design, assuming instead that end-users would have common requirements independent of their context. This may be a valid assumption for applications running on and accessed at desktop computers, but would be less appropriate for applications whose services are delivered via mobile devices [1, 9]. Ignoring the dynamic context of users may lead to suboptimal applications, at least for a subset of the context situations the end-user may find him/herself in. Therefore, especially driven by the successful uptake of mobile telephony and wireless communication, a new strand of applications has emerged, referred to as *context-aware* applications [12]. Such applications are, to a greater or lesser extent, aware of the end-user context situation (for example, “user is at home”, “user is travelling”) and provide the desirable services corresponding to the situation at hand. This quality points also to another related characteristic, namely that context-aware applications must be able to capture or be informed about information on the context of end-users, preferably without effort and conscious acts from the user part.

Developing context-aware applications is not a trivial task nevertheless and the following related challenges have been identified: (i) Properly deciding what physical context to ‘sense’ and what high-level context information to pass to an application, and bridging the gap between raw context data and high-level context information; (ii) Deciding which end-user context situations to consider and which to ignore; (iii) Modeling context-aware application behavior including ‘switching’ between alternative behaviors.

Inspired by the mentioned challenges, we have furthered a related work on context-aware application design [12], by explicitly discussing each of these interrelated challenges and proposing corresponding solution directions, supported by small-scale illustrative examples. It is expected that this contribution would usefully support the current efforts to improve context-aware application development.

The outline of the remaining of this paper is as follows: Section 2 further motivates the actuality of context-awareness as a desirable application quality. Section 3 provides relevant background information to be used as a basis for proposing improvements. Section 4, Section 5, and Section 6 address in more detail the challenges mentioned above, respectively. Finally, Section 7 presents our conclusions.

## 2 Motivation for Context-Awareness

The basic assumption underlying the development of context-aware applications is that end-user needs are not static, however partially dependent on the particular situation the end-user finds him/herself in. For example, depending on his/her current location, time, activity, social environment, environmental properties, or physiological properties, the end-user may have different interests, preferences, or needs with respect to the services that can be provided by applications.

Context-aware applications are therefore primarily motivated by their potential to increase user-perceived *effectiveness*, i.e. to provide services that better suit the needs of the end-user, by taking account of the user situation. We refer to the collection of parameters that determine the situation of an end-user, and which are relevant for the application in pursue of user-perceived effectiveness, as end-user context, or *context* for short, in accordance to definitions found in literature [4].

Context-awareness implies that information on the end-user context must be captured, and preferably so without conscious or active involvement of the end-user. Although in principle the end-user could also provide context information by directly interacting with the application, one can assume that in practice this would be too cumbersome if not impossible; it would require deep expertise to know the relevant context parameters and how these are correctly defined, and furthermore be very time consuming and error-prone to provide the parameter specifications as manual input.

Context-aware applications can be particularly effective if the end-user is *mobile* and uses a personal handheld device for the delivery of services. The mobile case is characterized by dynamic context situations often dominated by changing location (however not necessarily restricted to this). Different locations may imply different social environments and different network access options, which offer opportunities for the provision of adaptive or value-added services based on context sensitivity. Especially in the mobile case, context changes are continuous, and a context-aware application may exploit this by providing near real-time context-based adaptation during a service delivery session with its end-user. The adaptation is ‘near real-time’ because context information is an approximation (not exact representation) of the real-life context and thus there may be a time delay.

Through context-awareness, applications can be pro-active with respect to service delivery, in addition to being just re-active, by detecting certain context situations that require or invite the delivery of useful services which are then initiated by the application instead of by a user request. Otherwise said, traditional applications provide service in reaction to user requests (re-active), whereas context-aware applications have also the possibility of initiating a service when a particular context situation is detected, without user input (pro-active).

Although context-aware applications have received much attention within the research community, they have not been fully successful so far from a business point of view. This situation may change rapidly however, due to the observed growing power and reduced prices of mobile devices, sensors, and wireless networks, and due to the introduction of new marketing strategies and service delivery models [6,5].

In summary, context-awareness concerns the possibility of delivering effective *personalized* services to the end-user, taking into account his/her particular situation or context. Technological advances enable better and richer context-awareness, beyond mere location-sensitivity. Hence, service delivery models, specifically those targeting the mobile market, would allow companies to offer the added value in more attractive ways to the end-user.

Concerning the development and introduction of context-aware applications, as it has been mentioned already, this is not a trivial task. Efficiency and productivity would greatly benefit from an architecturally well-founded context infrastructure and design framework [17, 16, 3].

## 3 Architectural Implications and Design Considerations

In this section, we consider essential architectural/design issues concerning context-aware applications, and we also identify and briefly outline (on this basis) three important related interconnected challenges (to be elaborated in the following sections).

### 3.1 Architectural Implications

Context-aware applications acquire knowledge on context and exploit this knowledge to provide the best possible service. As already mentioned, the particular focus in this work concerns the end-user context, i.e. the situation of a person who is the potential user of services offered by an application. Examples of end-user context are the location of the user, the user's activity, the availability of the user, and the user's access to certain devices or facilities. The assumption we make is that the end-user is in different contexts over time, and as a consequence (s)he has changing preferences or needs with respect to services.

A schematic set-up for a context-aware application is depicted in Fig. 1. Here, the application is informed by sensors of the context (or of context changes), where the sensing is done as unobtrusively (and invisibly) for the end-user as possible. Sensors sample the user's environment and produce (primitive) context information, which is an approximation of the actual context, suitable for computer interpretation and processing. Higher level context information may be derived through inference and aggregation (using input from multiple sensors) before it is presented to applications which in turn can decide on the current context of the end-user and the corresponding service(s) that must be offered.

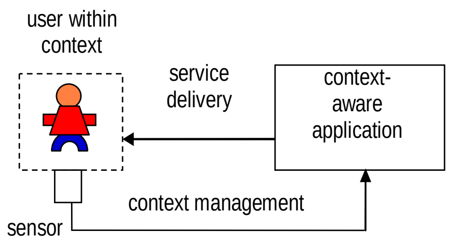


**Fig. 1.** Schematic representation of a context-aware application.

The design, implementation, deployment, and operation of context-aware applications have many interesting concerns, including:

- social/economical: how to determine useful context-aware services, where useful can be defined in terms of functional and monetary value?
- methodological: how to determine and model the context of the end-user that is relevant to the application; how to relate the context to the service of the application and how to model this service; how to design the application such that the service is correctly implemented?
- technical: how to represent context in the technical domain; how to manage context information such that it is useful to the application; how to use context information in the provisioning of context-aware services?

Addressing the last two concerns (especially the last one) starts with considering the possible architectures and in our view, two principle architectures could be proper:

- **Context-aware Selection**: end-user request(s) and end-user related context information are used to discover a matching service (or service composition). Discovery is supported by a repository of context-enhanced service descriptions. A context-enhanced service description not only specifies the functional properties (goals, interactions, input, output) and non-functional properties (performance, security, availability), but also the context properties of the service. Context properties indicate what context situations the service is targeting. For example, a service could provide information which is region-specific (such as a sightseeing tour), and therefore the context properties could indicate the relevance for a particular geographical area.
- **Context-aware Execution**: after the end-user request(s) has been processed and a matching service(s) has been found (possibly in the same way as described

above), the service delivery itself would adapt to changing context during the service session with the end-user. When the context of the end-user changes in a relevant (to the application) way, the service provided is adapted to the situation at hand. For example, the user may move from one location to another while using a service that offers information on objects of interest which are close-by (such as historic buildings within a radius of five kilometres, for example).

In both architectures, a new role is introduced, namely the role of *context provider*. A context provider is an information service provider where the information is context information. A context provider captures raw context data and/or processes context information with the purpose of producing richer context information which is of (commercial) interest. Interested parties could be other context providers or application providers. Further, a context-ware application obviously requires an *adaptive service provisioning component* and a *context information provisioning component*.

## 3.2 Design Considerations

Our design approach is a partial refinement of an existing approach [14] that concerns a general design life cycle, comprising, amongst others:

- **Business Modeling**: during this phase, the end-user is considered in relation to processes that either support him/her directly or the goal(s) of related business(es). These processes have to be identified, modeled and analyzed with respect to their ability to (collectively) achieve the stated goals. A model of these processes and their relationships is called a *business model*.
- **Application Modeling**: during this phase, the attention is shifted from the business to the IT domain. The purpose is to derive a model of the application, which can be used as a blueprint for the software implementation based on a target technological platform. A model of the application, whether as an integrated whole or as a composition of application components, is called an *application model*. Business models and application models should certainly be aligned, in order to achieve that the application properly contributes to the realization of the business/user goals. As a starting point for achieving proper alignment, one could delineate in the final business model which (parts of) processes are subject to automation (i.e., are considered for replacement by software applications). The most abstract representation of the delineated behavior would be a service specification of the application (as an integrated whole), which can be considered as the initial application model.
- **Requirements Elicitation**: both the business model and the application model have to meet certain requirements, which are captured and made explicit during the phase called *requirements elicitation*. Application requirements can be seen as a refinement of part of the business requirements, as a consequence of the proposition that the initial application model can be derived considering (parts of) the business processes (within the final business model), especially those processes selected for automation.
- **Context Elicitation**: an important part of the design of a context-aware application is the process of finding out the relevant end-user context from the

application point of view; we will refer to this phase as *context elicitation*. End-user context is relevant to the application if a context change would also change the preferences or needs of the end-user, regarding the service of the application. Context elicitation can therefore be seen also as the process of determining an end-user context state space, where each context state corresponds to an alternative desirable service behavior. Since relevant end-user context potentially has many attributes (location, activity, availability, and so on), a context state can relate to a complex end-user situation, composed of (statements on) several context attributes. Moreover, context elicitation relates to requirements elicitation in the sense that each context state is associated with requirements (i.e., preferences and needs of the end-user) on desirable user behavior. Context elicitation can best be done in the final phase of business modeling and the initial phase of application modeling, when the role and responsibility of the end-user and the role and responsibility of the application in their respective environments are considered.

Fig. 2 depicts these different phases and activities.

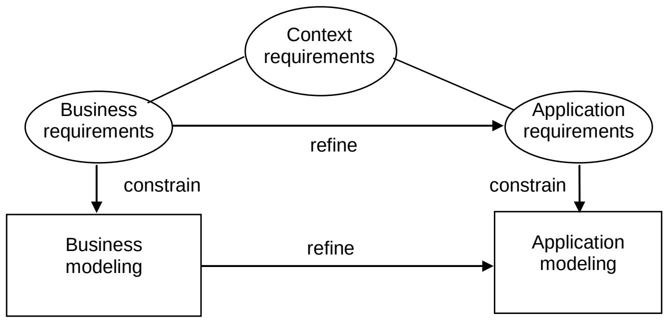


**Fig. 2.** Application design life cycle.

Following [12], we assume that an end-user context space can be defined and that each context state within this space corresponds to an alternative application service behavior. In other words, the application service consists of several sub-behaviors or variations of some basic behavior, each corresponding to a different context state. Any service behavior model would have to express the context state dependent transitions from one sub-behavior (or behavior variation) to another one.

## 3.3 Challenges

As mentioned already, developing context-aware applications is not a trivial task and the following related challenges have been identified:

- Properly deciding what to 'sense' and how to interpret it in adapting application behavior can be problematic since the interpreted sensed information must be a valid indication for a change in the situation of the end-user and it is not always trivial to know how context information is to correspond to a user situation.

- Deciding which end-user context situations to consider and which to ignore is challenging because there may be tens or even hundreds of possible end-user situations, with only several of them with high probability to occur, and therefore considering the others at design time is not sensible with respect to adequate resource expenditure.
- Modeling the application behavior including the ‘switching’ between alternative desirable application behaviors can be complicated because alternative behaviors are behaviors themselves which also are to be considered in an integrated way, allowing for modeling the ‘switching’ between them, driven possibly by rules.

In the following sections, we will address explicitly each of these challenges.

## 4 Deriving Context Information

An adequate decision about what should be ‘sensed’ and how it is to be interpreted, concerns the extraction of context information from raw data, which relates broadly to *context reasoning* [2].

Context reasoning is concerned with inferring context information from raw sensor data and deriving higher-level context information from lower-level context information. As for the extraction of context information from raw data, related algorithms are needed to support it, and two main concerns are to be taken into account:

- specific target applications, e.g. in domains such as healthcare or finance, requiring the *output* of the algorithms;
- the availability of sensors providing *input* to the algorithms.

Current standard mobile devices can already operate as sensors, e.g. they can gather GPS info, WiFi info, cellular network info, Bluetooth info, and voice call info. In addition, dedicated sensors (that for example measure vital signs) can be integrated with existing mobile networked devices. Next to that, future standard mobile devices may even include other types of sensors, e.g. measuring temperature.

Hence, it is considered crucial developing efficient context reasoning algorithms, by investigating whether it is possible to derive certain specific context information from certain specific sensor information. In order to adequately refine such algorithms, additional restrictions would need to be taken into account:

- restrictions concerning the (specific) processing environments of mobile devices;
- restrictions on memory usage, processing power, battery consumption, wireless network usage;
- restrictions that concern real-time versus delayed availability of extracted context.

In order to develop adequate algorithms that extract context from raw sensor data, it is thus important to appropriately consider gathering raw sensor data which is augmented with user input. Concerning the sensor data, it should be pre-processed and filtered, in order to be properly structured as input for the context reasoning algorithms which in turn would be expected to automatically yield the desired output. The (delivered) context information must be of certain (minimal) quality in order to be useful; otherwise said, certain Quality-of-Context levels should be maintained.

Finally, some issues that have more indirect impact, need also to be taken into account: (i) The delivered context information would have to be often applied in real-time environments where failures, performance requirements, available interfaces, and operational environments are to be taken into careful consideration; (ii) In order new applications to be enabled, it is important to investigate how the algorithms could be integrated in the infrastructure for context awareness.

## 5 Occurrence of Context Situations

Reasoning concerning context should point to the different situations the end-user may appear to be (situations that are characterized by corresponding context information. Often it is worthwhile considering the *occurrence probabilities* of these situations since, as mentioned already, usually only several (out of more) end-user situations are of high probability to actually occur. We call such an investigation *situation analysis*.

As studied in [12], it is helpful to support such an analysis by means of ‘pragmatic’ decisions (for instance: to ignore end-user situations which usually do not occur, although they might occur with some (certainly small) probability). Such subjective decisions should however be rooted in more objective studies that justify the decision(s) taken. In our view, a possible way of approaching this is through *random variables*. Exploring their probabilities allows one to apply statistical analysis, including *hypotheses testing* and *parameters estimation* [7].

Considering just possible outcomes is sometimes not enough in approaching a phenomenon; one might need to refer to an outcome in general. This is possible through a random variable, if the occurrence probability of the outcomes is studied (a random variable is a function that associates a unique numerical value with every outcome of an experiment).

An experimental data bank could be built through observations. Then, by applying statistical analysis, the development team would get the right insight on: (i) which end-user situation to be defined as the ‘default’ situation (the situation that points to the ‘default’ application behavior); (ii) which of the other situations are to be put ‘for consideration’; (iii) which (obviously the rest) should be ignored. This is illustrated in Fig. 3 (where **n** should be certainly equal to **m**+**p**+**1**):

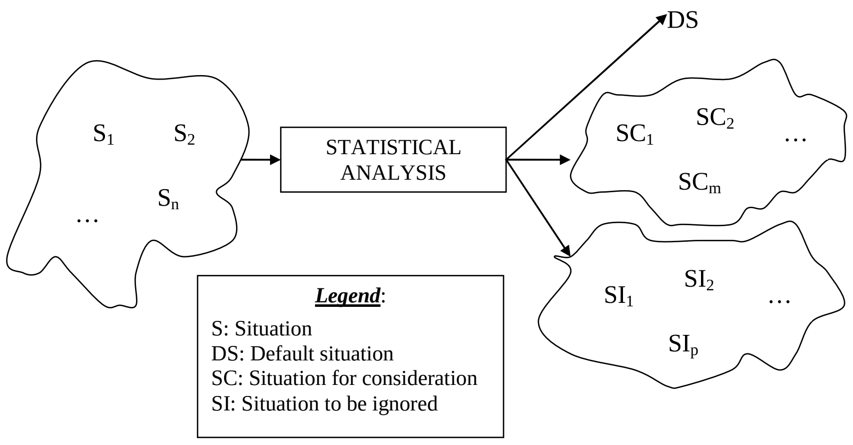


**Fig. 3.** Applying statistical analysis.

In a healthcare-related example, considered in [12], a hospital could be viewed as an end-user and there are exactly two possible end-user situations or states (considered as possible outcomes), namely: 'not too busy' (some medical doctors are immediately available to provide help) and 'very busy' (all medical doctors are occupied or have scheduled appointments within half an hour, for example). We consider the random variable $\mathbf{Y}$ with respect to these outcomes. $\mathbf{Y}$ would be a discrete random variable [7] since it may take on only a countable number of distinct values (in our case two). Provided the number of possible distinct values is exactly two, we have the case of *a priori probabilities* of each of the alternative outcomes (this means that one of these probabilities can be calculated by deducting the other one from $\mathbf{1}$).

Only for the purpose of exemplifying how statistical analysis (applied to information that has been collected through observations) could be of use for the application designer, we take the probabilities from the mentioned example: the a priori probability of the first of the mentioned possible outcomes ("not too busy") is $\mathbf{0.9}$ and the a priori probability of the second alternative outcome ("very busy") is therefore $\mathbf{0.1}$.

Knowing the occurrence probability of each outcome helps in deciding (in this particular example) which to be the 'default' desirable application behavior (the other one – that points to the other alternative outcome – would be the alternative behavior). It would be of course sensible considering the application behavior that corresponds to the first possible outcome as the 'default' behavior.

Once the designer has grouped the possible end-user situations, as suggested by Fig. 3 (only a 'default' and 'alternative' situations to be considered in the example), it is important making sure that the application is capable of 'sensing' the end-user situations. The proposed way of solving this is through observation of the values of appropriate *parameters*. If there are $\mathbf{n}$ parameters relevant to a scenario, then each of the parameters would have certain possible *values*. Then each value combination would point to a particular end-user situation.

In the example, we might distinguish two parameters ($\mathbf{p}_1$ and $\mathbf{p}_2$) and five corresponding values, as follows:

- $\mathbf{p}_1$ is about the ratio between the number of patients and the number of medical doctors at the particular moment, and is with just three possible values: $\mathbf{v}_{11}$ (the number is less than **1**), $\mathbf{v}_{12}$ (it is exactly **1**), and $\mathbf{v}_{13}$ (it is more than **1**)
- $\mathbf{p}_2$ concerns the particular moment – *normal* (the hospital is supposed to function as usual during working hours) or *extreme* (the hospital can rely on limited (human) resources, as during night-time, for example), and has just two possible values, respectively for normal and extreme, namely $\mathbf{v}_{21}$ and $\mathbf{v}_{22}$.

There are six possible value ($\mathbf{p}_1$,$\mathbf{p}_2$) combinations, namely $\mathbf{v}_{11}.\mathbf{v}_{21}$, $\mathbf{v}_{11}.\mathbf{v}_{22}$, $\mathbf{v}_{12}.\mathbf{v}_{21}$, $\mathbf{v}_{12}.\mathbf{v}_{22}$, $\mathbf{v}_{13}.\mathbf{v}_{21}$ and $\mathbf{v}_{13}.\mathbf{v}_{22}$. Driven by some additional domain analysis, omitted here for brevity, we determine the last combination only as validly corresponding to the ***0.1**-probability* alternative (the 'Second' alternative), and thus all the rest, corresponding to the ***0.9**-probability* alternative (the 'First' alternative), as depicted in Fig. 4.

| | Parameters' values' combinations |
|---|---|
| First alternative | $v_{11}.v_{21}$, $v_{11}.v_{22}$, $v_{12}.v_{21}$, $v_{12}.v_{22}$, $v_{13}.v_{21}$ |
| Second alternative | $v_{13}.v_{22}$ |

**Fig. 4.** Recognition of end-user situations.

Hence, knowing the values of the two parameters (the values can usually be captured using sensors), one could actually 'sense' the end-user situation at a particular moment [12].

## 6 Managing Alternative Application Behaviors

After a consideration of the different possible end-user situations that point to (corresponding) alternative application behaviors, the application designer has to adequately address the challenge of managing these behaviors; even though the 'switching' between behaviors would take place at real time, proper design time preparations are to be realized. These preparations should not only concern the modeling of each of the alternative behaviors to be considered but they should also address the 'switching' between behaviors (driven by a change in the end-user situation).

Taking into account that the 'switching' between alternative behaviors is insufficiently elaborated in current approaches [11,12] and inspired by previous experience, we propose the usage, in combination, of *Petri Net* [15] and *Norm Analysis* [8,13].

Petri Net could be considered as a triple (**P**,**T**,**F**) that consists of two node types (*places* and *transitions*), and a flow relation between them. Places are to model milestones reached within a business process and transitions should correspond to the individual tasks to execute. Places are represented by circles, transitions are represented by rectangles. The process constructions which are applied to build a business process, are called *blocks*. They express some typical constructs, such as sequence, choice, parallelism, and iteration. Hence the strengths of Petri Net, concerning the modeling of decision points and parallel processes, are especially relevant to the challenge of modeling alternative behaviors. Using the same notations for conveniently modeling at different abstraction levels, gives the precious possibility to grasp the 'big picture' and go consistently in details, and also to map to other notations, and also to simulate. A further challenge nevertheless that concerns not only Petri Net but also other process modeling formalisms, is the insufficient elaboration facilities with regard to 'decision' and other complex points. We claim that combining Petri Net and Norm Analysis (to be introduced further in the current section) could be a good solution in this perspective [10].

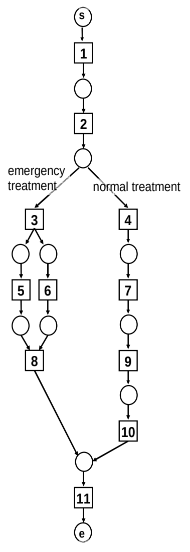
Norm Analysis essentially concerns Semiotic Norms, or *norms* for short, which include formal and informal rules and regulations, define the dynamic conditions of the pattern of behavior existing in a community and govern how its members (agents) behave, think, make judgements and perceive the world. When the norms of an organization are learned, it would be possible for one to expect and predict behavior and to collaborate with others in performing coordinated actions. Once the norms are understood, captured and represented in, for example, the form of deontic logic, this could serve as a basis for programming intelligent agents to perform many regular activities. The long established classification of norms is probably that drawn from social psychology, partitioning them into perceptual, evaluative, cognitive and behavioral norms; each governing human behavior from different aspects. However, in business process modeling, most rules and regulations fall into the category of behavioral norms. These norms prescribe what people must, may, and must not do, which are equivalent to three deontic operators: "of obligation", "of permission", and "of prohibition". Hence, the following format is considered suitable for specification of a behavioral norm:

*whenever <condition>*
*if <state>*
*then <agent>*
*is <deontic operator>*
*to <action>*

The condition describes a matching situation where the norm is to be applied, and sometimes further specified with a state-clause (this clause is optional). The actor-clause specifies the responsible actor for the action, who could be a staff member, or a customer, or a computer system if the right of decision-making is delegated to it. As for the next clause, it quantifies a deontic state and usually expresses in one of the three operators - permitted, forbidden and obliged. For the next clause, it defines the consequence of the norm. The consequence possibly leads to an action or to the generation of information for others to act. With the introduction of deontic operators, norms are broader than the normally recognised business rules; therefore provide more expressiveness. For those actions that are "permitted", whether the agent would take an action or not is seldom deterministic. This elasticity characterises the business processes, and therefore is of particularly value to understand organisations.

The combination between Petri Net and Norm Analysis is of interest, especially with regard to the challenge of managing alternative application behaviors, for a number of reasons, among which are the following:

- Petri Net is a well-established process modeling formalism with sound theoretical roots and 'convenient' notations, that only misses facilities for exhaustive elaboration concerning complex points, while Norm Analysis is a well established rule modeling formalism possessing also sound theoretical roots and impressive (process-elaboration-related) expressiveness.
- There are examples of applying Petri Net and Norma Analysis in combination [10].
- The useful capability of modeling and elaborating (through Petri Net + Norm Analysis) complex process constructs makes the Petri Net – Norm Analysis combination attractive particularly with regard to the challenge of managing alternative application behaviors.



**labels of transitions**

s: start

1: register patient

2: check emergency status

3: list patient in 'traffic-light' (TL) system

4: list patient in a queue

5: examine vital signs of patient

6: check health history of patient

7: analyze record of patient

8: prescribe emergency treatment

9: examine patient

10: formulate diagnosis

11: treat patient

e: end

**whenever** a patient needs emergency help
**then** the receptionist
**is** obliged
**to** list the patient in the TL system.

**whenever** a patient does not need emerg. help
**then** the receptionist
**is** obliged
**to** list the patient in a normal queue.

**Fig. 5.** A typical health-care process.

Fig. 5 (left) presents a typical health-care process, using Petri Net, and it is easily seen that there are two alternative behaviors, namely emergency and normal treatment. We could use Norm Analysis in such cases to usefully elaborate the process model. For instance, two norms corresponding to the choice construct in Fig. 5 (left) can be identified and specified in detail – consider Fig. 5 (right).

Therefore, by combining Petri Net and Norm Analysis, one could substantially facilitate the handling of (alternative) application behaviors.

## 7 Conclusions

This paper has presented further results that concern the development of context-aware applications. In particular, following a related motivation statement and based on architecture/design visions on the development of context-aware applications, we have identified and outlined three related interconnected challenges, proposing and motivating afterwards corresponding solution directions, summarized as follows:

- To decide what to 'sense' and how to interpret it in adapting application behavior, one would need to apply *context reasoning* for the purpose of properly extracting context information from raw data (the guidelines presented in Section 4 could be useful in this direction).
- To decide (at design time) which should be the 'default' application behavior, which alternative behaviors to remain under consideration, and which behaviors may be ignored, one could get useful support through analyzing (considering random variables) the occurrence probabilities of end-user situations; on the basis of observations, statistical analysis can be applied in support of such decisions. As for the 'sensing' the end-user situations corresponding to these application behaviors, one could consider observing the values of appropriate parameters (the guidelines presented in Section 5 could be useful in this direction).
- To appropriately model the complex behavior of a context-aware application including 'switching' between alternative behaviors, one would require not only a powerful process modeling formalism but also an appropriate elaboration facility to be applied to complex points (the proposed in Section 6 combined application of Petri Net and Norm Analysis could be useful in this direction).

It is expected that these results would usefully support the current efforts to improve context-aware application development

However, all addressed challenges and corresponding solution directions must be considered in an integrated manner, as part of a context-aware application development approach, since they are interrelated. Hence, we plan (as further work) to use the results reported in the current papers for extending usefully an existing business-application-alignment approach [12].

## Acknowledgements

This work is part of the Freeband A-MUSE project (http://a-muse.freeband.nl). Freeband is sponsored by the Dutch government under contract BSIK 03025.

## References

1. Alonso, G., Casati, F., Kuno, H., Machiraju, V.: Web Services, Concepts, Architectures and Applications. Springer-Verlag, Berlin-Heidelberg (2004)

2. AWARENESS, Freeband AWARENESS project, http://www.freeband.nl/ project.cfm?id= 494&language=en (2008)
3. Broens, T.H.F., van Halteren, A.T., van Sinderen, M.J., Wac, K.E.: Towards an Application Framework for Context-Aware m-Health Applications. International Journal of Internet Protocol Technology, 2 (2) (2007)
4. Dey, A. K.: Understanding and Using Context. Personal Ubiquitous Computing 5 (1): 4-7 (2001)
5. Hristova, N., O'Hare, G.M.P.: Ad-me: Wireless Advertising Adapted to the User Location, Device and Emotions. In: HICSS'04, 37th Hawaii International Conference on System Sciences (2004)
6. Kurkovsky, S., Harihar, K.: Using Ubiquitous Computing in Interactive Mobile Marketing. Pers Ubiquit Compt, vol. 10, no. 1 (2006)
7. Levin, R.I., Rubin, D.S.: Statistics for Management. Prentice Hall, USA (1997)
8. Liu, K.: Semiotics in Information Systems Engineering. Cambridge University Press, Cambridge (2000)
9. Schilit, B., Adams, N., Want, R.: Context-Aware Computing Applications. In: WMCSA'94, Workshop on Mobile Computing Systems and Applications (1994)
10. Shishkov, B. and Dietz, J.: Deriving Use cases from Business processes, the Advantages of DEMO. *In: Enterprise Information Systems V.* Eds. O. Camp, J.B.L. Filipe, S. Hammoudi, and M. Piattini. Kluwer Academic Publishers, Dordrecht/Boston/London (2004)
11. Shishkov, B., Quartel, D.: Combining SDBC and ISDL in the Modeling and Refinement of Business Processes. *In: Enterprise Information Systems VIII, Eds.: Y. Manolopoulos, J. Filipe, P. Constantopoulos, J. Cordeiro, Lecture Notes in Business Information Processing.* Springer-Verlag, Berlin-Heidelberg (2008)
12. Shishkov, B., Van SInderen, M. J.: From User Context States to Context-Aware Applications. *In: Enterprise Information Systems IX, Eds.: J. Cordoso, J. Cordeiro, J. Filipe, V. Pedrosa, Lecture Notes in Business Information Processing.* Springer-Verlag, Berlin-Heidelberg (2008)
13. Shishkov, B., Dietz, J.L.G., Liu, K.: Bridging the Language-Action Perspective and Organizational Semiotics in SDBC. In: ICEIS'06, 8th International Conference on Enterprise Information Systems (2006)
14. Shishkov, B., Van Sinderen, M.J., Quartel, D.: SOA-Driven Business-Software Alignment. In: ICEBE'06, IEEE International Conference on e-Business Engineering (2006)
15. Van Hee, K.M., Reijers, H.A.: Using Formal Analysis Techniques in Business Process Re-Design. *In: Business Process Management; Models, Techniques, and Empirical Studies, Eds.: W. van der Aalst, J. Desel, A. Oberweis, Lecture Notes in Computer Science.* Springer-Verlag, Berlin-Heidelberg (2000)
16. Van Sinderen, M.J.: Architectural Styles in Service Oriented Design. In: ICSOFT'06, International Conference on Software and Data Technologies (2006)
17. Van Sinderen, M.J., Van Halteren, A., Wegdam, M., Meeuwissen, E., Eertink, H.: Supporting Context-Aware Mobile Applications: An Infrastructure Approach. IEEE Communications Magazine 44 (9): 96-104 (2006).

# Utility Computing Paradigm and SOA Philosophy

Ivan Ivanov

Empire State College, State University of New York, Long Island Center
Hauppauge, NY 11788, U.S.A.
Ivan.Ivanov@esc.edu

**Abstract.** The purpose of this paper is to sort out, to the extent possible, the contentious discussion regarding the impact of service oriented architecture to utility computing. How useful is this philosophy in conjunction with utility computing approaches on organizational IT strategies, business processes and directional models? The change to IT utilization is being driven by the infrastructural advantage and economic leverage of the Internet in combination with imperative industry trends: commoditization of IT, Service-Oriented Architectures (SOA) and Virtualization of Services and Applications. These trends include several distinct innovations such as:

- the use of multiple servers to replace large expensive systems (IT commoditization);
- the componentization of flexible application building blocks that can be easily assembled into large, composite business specific applications (Service Oriented Architectures);
- the virtualization of operating systems, data storage, network resources, computing power (grid computing) and applications (as a top layer of virtualized services).

The business approach seems to achieve the transformation of IT from an inert monolith to a dynamic, business adaptive model. This forms the Utility Computing paradigm. However, the question remains, how well do the UC models synthesize with the agility provided by SOA philosophy to enable a continuous optimization of business processes?

## 1 Introduction

Information Technology (IT) has had a profound impact on organizational strategies, infrastructure, services, and business models in the last several decades. The transformations inside IT industry as applications, services, and solutions have been changing persistently because of customers' needs and business necessities.

In the early stages in the development of information and communication technology, there are few standards, majority proprietary (company specific) products and solutions, limited applications, and deficient distributed network; as a result, IT has been impossible to provide expected economies. IT solution and services were usually regionally dependent and fragmented by product and applications. Such fragmentation has appeared intrinsically lavish for the businesses. It compelled large capital investments, heavy fixed IT expenses: both in the technology and in operational costs (administration, monitoring, and maintenance), resulting in high levels of overcapacity. The situation has been ideal for the suppliers of the technology components and infrastructural builders, but it has been ultimately unsustainable.

The economic difficulties in 2000s, the cost-effective strength of the Internet and some new technological advantages have made businesses more vigilant and more demanding about the return of their IT infrastructural investments. As a crucial business resource IT has matured and came to what economists describe as a General-Purpose Technology (GPT), sharing four specific characteristics of GPTs:

- Wide scope for improvement and elaboration,
- Applicability across a broad range of uses,
- Potential for use in a wide variety of products and processes,
- Strong complementarities with existing or potential new technologies [4].

Because of the broad range of employments, variety of products and applications, IT as a typical GPT proffers the potential of considerable economies of scale if their supply can be unified and consolidated. The business approach seems to achieve the transformation of IT from an inert monolith to a dynamic organism, better adaptive to the business needs model. While delivering IT as utility has been recognized and a central distribution becomes possible, large-scale utility suppliers arise and displace the smaller product-specific providers. Although companies may take years to abandon their proprietary supply IT operations and all the sunk costs they represent, the savings offered by utilities eventually become too compelling to resist, even for the largest enterprises[3].

The transformation to IT utilization is being driven by the infrastructural advantage and economic leverage of the Internet in combination with imperative industry trends that advance and permit realization of over-the-net different delivery models. These trends include several distinct innovations such as:

- the use of multiple servers to replace large expensive systems (IT commoditization);
- the componentization of flexible application building blocks that can be easily assembled into large, composite business specific applications (Service Oriented Architectures);
- the virtualization of operating systems, data storage, network resources, computing power (grid computing) and applications (as a top layer of virtualized services).

The purpose of this paper is to sort out, to the extent possible, the contentious discussion regarding the impact of service oriented architecture approach to utility computing models. The rest of the paper is structured as follows: Section 2 - Utility Computing Paradigm expose the concept, technologies, models and the paradigm sifts for consumers, vendors and providers of utility computing; Section 3- SOA Philosophy and IT agility-integration reveals how SOA approaches can be deployed to achieve agile business integration; Section 4 - The Implication of SOA within UC structures illustrates some developments in employing SOA approaches within hybrid utility computing models to advance integration of existing and newly developed applications and to attain extensive economies; Section 5 – Conclusion, ends the paper with closing notes.

# 2 Utility Computing Paradigm

## 2.1 The Concept

Utility computing was first described by John McCarthy at the Dartmouth conference in 1955 as: "If computers of the kind I have advocated become the computers of the future, then computing may someday be organized as a public utility just as the telephone system is a public utility… The computer utility could become the basis of a new and important industry." The major factors which impeded the development of computer utilities in the last decades were:

- high data communications costs,
- timid public attention,
- limited number of trained and skilled IT users,
- lack of standardization of hardware, software and data communications,
- apprehensive compilation of database systems and development tools,
- high level of security threats.

Practically fifty years were needed to develop a broad-spectrum of computerized devices, universal communication infrastructure and over-the-net applications, to saturate organizations and users with appropriate computer systems and more adaptive technology solutions. This time period was vital to educate a critical mass of IT professionals in programming, networking, business productivity systems and web based applications, and to train a vast majority of end-users how to utilize them [10].

## 2.2 Utility Computing Technologies

The recent utility computing development as a complex technology involve business procedures that profoundly transform the nature of companies' IT services, organizational IT strategies, technology infrastructures, and business models. Based on networked businesses and widely implemented Over-the-Net applications, utility computing facilitates "agility-integration" of IT resources and services within and between virtual companies.

There is immense variety in possible and actual configurations of technologies and infrastructure to support utility computing development. According to Alfredo Mendoza [12], well established and proven technologies like virtualization, advanced application accounting, and dynamic partitioning, that have long existed in mainframes and now are available on newer server architectures in combination with grid computing, web services and hyperthreading technologies are contributing to create an infrastructure based on the utility model. Other experts believe utility computing will further evolve into a combination of the related concepts of grid computing (a type of network-distributed parallel processing), on-demand, and Web services [18]. The primary newly established technologies for companies seeking a competitive advantage in utility computing development are grid computing, all forms of virtualization services and automated provisioning.

### 2.2.1 Grid Computing

In a grid, all of the networked computers are coordinated and act as a single “virtual” computer. Grids use specialized scheduling software that identifies available resources and allocates tasks for processing accordingly. "A grid cluster is a collection of independent machines connected together by a private network with a specific software layer on top. This software layer has to make the entire cluster look like a single computing resource." -- Don Becker, CTO, Penguin Computing (a manufacturer of Linux-based grid solutions), offers a succinct definition of grid computing.

The key element is that computers, or nodes, in a grid are able to act independently without centralized control, handling requests as they are made and scheduling others. Grid computing is the underlying technology for utility computing. In a long term, grid computing is heading towards a convergence of utility computing from the pricing and delivery prospective, and Web services-based integration and virtualized technologies to enable multiple, networked computers to be managed as one [17]. Amongst systems vendors developing and exploiting grid concepts are HP with HP Adaptive Enterprise Initiative, Sun Microsystems Network One, IBM’s On-Demand Computing, and Oracle Grid Computing.

The grid may increase geographically in organizations that have facilities in different cities and continents. Dedicated communications connections, VPN tunneling and other technologies may be applied among different parts of organizations and the grid. The grid may grow to be hierarchically organized to reduce contention implied by central control, while increasing scalability. With developing the grid infrastructure, the grid may expand to cross organization boundaries migrating to “Intergrid”, and may be used to collaborate on projects to provide brokering and trading resources over a much wider audience; those resources may be then purchased as a utility from trusted suppliers.

### 2.2.2 Virtualization

Virtualization services allow servers, storage capacity, network resources or any virtual application to be accessed and referenced independent of its physical characteristics and location. Virtualization presents a logical grouping or subset of computing resources such as hardware, operating systems, storage and applications, which may be accessed to enhance the original configuration. The improvement with virtual resources is not limited geographically, by applications, or physically, such as in configuration. Solution providers can use server virtualization and other virtual appliances to provide new services. Server virtualization is used to create utility computing server farms that combine multiple customers' workloads. The cost-to-customers is based on metrics, such as the gigabytes of memory and disk space used, computing power or servers needed. This maximizes the customers' ROI with a pay-as-you-go model. It also allows access to an infrastructure, which operates on-demand. A server farm can be used to duplicate or expand, rather than replace, a customer's infrastructure. This may become important if a natural disaster should happen, for instance, requiring migration of images from the customer's servers to laptops or another system [15].

Stating it succinctly, virtualization for most vendors specialized in this technology is an abstract layer that allows multiple virtual machines, with heterogeneous operating systems to execute in separation side-by-side on the same physical system. Virtualized services allow customers to utilize and expand their systems in many directions such as:

- Server consolidation - combine many physical servers into fewer, highly scalable enterprise-class servers, which host virtual machines, also known as physical-to-virtual (P2V) transformation.
- Storage virtualization – high-speed data-storage switched networks, such as Storage Area Networks (SAN) and Network-attached Storage (NAS), provide shared access to many storage devices, virtual file servers or file systems.
- Network virtualization – segregates the inbuilt network resources into separate, distinct and secure channels and devices composing virtual private networks (VPNs), “demilitarized zone” in the context of firewalls, load balancers and voice over IP services.
- Disaster recovery and business continuity protection - alters historical backup-and-restore (virtual machines are used as "hot standby" environments, which allow backup images to migrate and "boot" into live virtual machines).
- Streamline Testing and Training - hardware virtualization allows root access to a virtual machine that is useful in kernel development, operating system training and application testing.
- Portability for Applications and Automation Capabilities - applications virtualized for portability will remain portable, while virtual appliances combine simple deployment of software with the benefits of pre-configured devices.
- Streaming Applications and Secure Enterprise Desktops - virtualized software locked down onto the local Desktop, by providing a standard corporate desktop image in a virtual machine, while the standardized desktop enterprise environment is hosted in virtual machines accessed through thin clients or PCs.

VMware, is one of the leading providers for virtualization technology systems. As said by VMware president Diane Greene “Once you aggregate your hardware resources, you can allocate a certain amount of CPU power, memory, disk and network to a group of virtual machines, and it will be guaranteed those resources. If it’s not using them; other virtual machines will be able to use those resources… It’s utility computing made real and working” [7]. Recently launched Virtual Application Environment by Microsoft provides application extensive virtualization that can be layered on top of other virtualization technologies – network, storage, machine – to create a fully virtual IT environment where all computing resources can be dynamically allocated in real-time based on real-time needs. Applications are turned into on-demand utilities that can be used on any system, easy to dynamically add, update and support, creating nimble business environment, using minimal time and resources [13]. Virtualization techniques might affix a little higher operating costs and complexity compared to nonvirtual settings, but there many other capabilities and advantages of having virtualized resources that will bring much higher economies and reliability.

### 2.2.3 Provisioning

Utility computing is generally a provisioning model - its primary purpose is to only provide a service when, how, and where it is needed. Automated or manual provisioning of resources in a large scale provides access to new servers or additional capacity in an automated and “on-the-fly” manner. Since utility computing systems create and manage many, and simultaneous, occurrences of a utility service, each one providing application functions, it becomes necessary to establish provisioning policies. The Internet Engineering Task Force (IETF) has adopted a general policy-based administration framework with four basic elements: (1) a policy management tool, (2) a policy repository, (3) a policy decision point, and (4) a policy enforcement point.

The market and technology leader in this technology trend IBM, has implemented three main categories of policies related to the provisioning of services within a utility computing system:

- the service provider (SP), who deal with the sharing of the computing infrastructure among different on-demand services (ODS),
- the utility computing service environments (UCSE), which deal with policies associated with the allocation and management of computing resources supporting a given ODS, and
- the resource managers, who deal with the administration of pools of specific resources.

The type of provisioning provided depends upon the utility model implemented. For a storage area network (SAN), for example, provisioning involves assigning process space to optimize performance. IBM's on-demand architecture considers each instance of a utility service a "utility computing service environment" (UCSE). Recently, many companies are retooling their infrastructure to incorporate virtualization technologies to work with policy-based automation management software geared toward automated provisioning. The increasing need for more flexible IT services will gear to a more consolidated and automated infrastructure environment [12].

The above described utility computing technologies are supported by further advances – the increased deployment of blade servers, inexpensive high-speed networks development, the adoption of open source technologies and software as a service approach, evolving policy-based automation and application management software to streamline over-the net application allocation and management. With their modular uni-, dual- or multiprocessor architecture, blade servers offer tremendous space saving, solid performance and easy of management environment. All these tendencies sit well with virtualization, grid computing and the allocation of computing resource on-the-fly.

## 2.3 Utility Computing Model and the Paradigm Shift

The term "utility computing" is still pretty new and the phrase generates confusion, since it is commonly used to describe a technology as well as a business model. The difficulty is that computing is not nearly as simple as conventional utilities. Computing involves a vast amount of context, as opposed to volts, amps and watts for the most complex other public utility - the electricity. The utility computing uniquely integrates storage, applications, computational power and network infrastructure as a foundation for business adjustable IT services. In the ultimate utility computing models, organizations will be able to acquire as much IT services and applications as they need, whenever and wherever they need them.

Utility computing is a model that allows breaking down IT infrastructure into discrete pieces that can perform different and separate business functionalities, can be measured independently, can be turned on and off as necessary [12]. It offers companies and private users an access to hosted computing services, scalable and portable business applications through a utility-like, pay-on-demand service over the Internet network. To achieve cost savings, to reduce IT complexity and to increase IT flexibility and integration ability when utility computing model is being applied, suppliers and consumers of utility services need to reach a higher level of standardization and sharing. The five levels of Continuum of Utilities model, illustrated by Alfredo Mendoza in Utility Computing: Technologies, Standards, and Strategies, exposes some critical developments and infrastructural transformations towards approaching a higher level of standardizations, consolidations and sharing:

- Level 1 - *Utility Pricing* – New technology enables utility like functionalities and pricing in services within the infrastructure. Typical examples are: capacity on demand, on-demand computing, pay-per-use, pay-per-service where utility suppliers and consumers specify the scope and the frame of computing services and negotiate the utility pricing model
- Level 2 – *Infrastructure Utility* – At this level, new technologies such as virtual servers, storage and networks with advanced partitioning, automated provisioning and policy-based management facilitate processes of virtualized operating environment and allocate resources as and where needed
- Level 3 – *Shared Application utilities* – Architectural changes to enterprise software applications derived from Service-oriented architecture (SOA) implemented as Web services, and metering Software as a Service (SaaS) for enterprise applications transform single instance applications into multi-tenant applications served over-the-net
- Level 4 – *Shared Process Utilities* – At this level, companies identify business functionalities that are non-strategic, deconstruct to similar functional components within different processes to externalize or shared with other organizational entities within the networked environment
- Level 5 – *Virtual Infrastructure Utilities* – At the last most advanced level infrastructure utilities begin to share resources with each other. Communications between utilities are done through industry-standard protocols and format such as data communications markup language (DCML) and are possible by sharing resources between separate data centers through the use of grid infrastructure or utility computing environment.

The utility computing model creates a substantial magnitude in the paradigm shift for vendors, providers and consumers of computing power and IT services. Risk-reluctant organizations would take more discrete phase based approach when applying some utility like services without affecting critical business systems. Imaginative providers will reflect the new paradigm by offering a variety of utility-based options - from specific customized systems through hybrid stepwise services to total utility solutions. In the following paragraphs are listed some of the key steps and techniques companies switching to the new utility computing paradigm should consider from consumers' and providers' prospective. According to Gardner Group study the utility computing suppliers are going through five stages to build their utility infrastructure: (1) concentration of resources, (2) consolidation of assets, including infrastructure facilities, (3) virtualization of services, (4) automation of processes, and (5) extension of services and solutions. Firms move from one stage to the next, with each stage firmly established before going to the next [6].

The leading companies in the utility paradigm recently are in late stage 3 or stage 4; they make available a wide range of automation processes and business operations deployment based on virtualized computing resources and services. Sun with its N1 architecture, Grid compute utility and StorEdge services provide virtualization of data center resources, dynamic allocation of IT applications, automation of installation, configuration, accounting and reporting on per-service basis deployment. HP Adaptive Enterprise is the HP shift to utility computing development. The HP strategy is to deliver virtualization technology and utility computing services at different product levels: individual or element-based virtualization, integrated virtualization, and metered, managed and instant capacity operations. While the diversity of utility like options is substantial and HP is acting as a typical IT utility provider, there are some strategic HP advances in servers, storage, imaging and printing services. In 2006, HP won two multiyear $440M utility computing contracts from the United States Federal Government. Based on its worldwide communication network, specialized services and cross-platform expertise, HP deploys adaptive infrastructure using HP Integrity and HP ProLiant servers, and delivers software solutions for automated server provisioning, configuration, patch and IT asset management. HP discontinued its Utility Data Center (UDC) monolith initiative advancing more flexible and granular utility computing services such as imaging and printing operations, server and storage virtualization and automated provisioning on modular platforms to target larger costumers' groups and variety of business expectations.

IBM On Demand strategy is the company's complex utility computing model, which incorporates infrastructure virtualization and management technologies, application hosting services and business process operations. IBM has proved its leading expertise in this realm with many successful utility projects from modular business specific applications to the most comprehensive IT solution to American Express announced in late 2002. "Today American Express is placing itself at the forefront of a new computer services paradigm," said Doug Elix, IBM senior vice president and group executive, IBM Global Services. "The utility computing service delivery model American Express is adopting will give it the flexibility to draw on all the computing resources, skills and technologies required to support future growth." The agreement saves American Express hundreds of millions of dollars in information technology costs, and having IBM's resources on demand provides AmEx with the flexibility to adjust rapidly to changing business needs.

The pragmatism that drives most organizations as consumers into utility model is not only immediate cost savings, but also how IT is structured and managed, accounted for, and used to enable businesses to improve their efficiency and effectiveness. In today's world, IT differentiation in products or services is unlikely to be achieved; therefore more executives are looking to business process innovation as a key competitive advantage. Virtually all businesses could take advantage and building out a company-specific platform by employing best pieces of proved utility computing options in different timeframe [10]. The timeframe IDC envisages regarding the major steps customers would advance when they incorporate utility principles methodically and incrementally includes four phases: (1) Virtualization 1.0 – Encapsulation, Resource Sharing and Dynamic Consolidation – 2005, (2) Virtualization 2.0 – Mobility and Planned Downtime - 2007, (3) Virtualization 2.5 – Unplanned Load, Alternate Disaster Recovery Workload Balancing – 2009, and (4) Virtualization 3.0 – Automation Provisioning, Service Oriented and Policy Based solutions, Variable Costs – 2010+ [8].

The important strategic decision consumers must take is the type of computing utility: *private* (in-house) utilities, *public* utilities or *hybrid* (selective) utilities. The answer depends on existing IT resources, infrastructure and professional expertise the company possesses. Organizations could initiate small pilot projects, examine new utility type services, and build expertise and confidence with implementation of new technologies supporting utility computing paradigm. According to leading IT research institutions (Gartner, Forrester, IDC) the operational costs are between 55 to 75 % of the total IT costs and they are growing at twice the rate of overall expenses. In 2004, IDC reported $55 billions are located for buying new servers and $95 billions to manage them, while for 2008 new servers spending is expected to reach $60 billions but the management cost would rise to around $140 billions. Employing utility computing services, organizations could expect 30-65% decrease in operational costs and over 50-75% savings from total cost of ownership.

## 3 SOA Philosophy and it Agility-Integration

Service-Oriented Architecture is a software design approach that dissolves business applications into separate functions or “services” – e.g. check credit history, or open new account – that can be used independent of the applications and computing platforms on which they run. When individual functions within applications are all available as discrete building blocks, companies have the ability to integrate and group them differently to create new capabilities and align to business processes [9]. This architectural approach is specifically applicable when multiple applications and process running on various technologies and platforms need to interact with each other – a recurring scenario within utility computing environment.

SOA is a logical way of designing a software system to deliver services to either end-user applications or other services distributed over-the-net through published and discoverable interfaces. The basic SOA defines an interaction between software agents as an exchange of messages between service client and service provider, while both parties - provider and client - are respectively responsible for publishing a description of the service(s) they provide and finding description of service(s) they require and they must be able to bind them [14]. The SOA offers a model in which relatively loosely coupled collections of existing IT assets (called services) are reused and reconnected providing the functionality required by business applications, management functions, and infrastructure operations. In this way, supply chain partners and value nets can more easily integrate business processes and applications across organizational boundaries and achieve better integration, greater flexibility, and improved ease of cooperation and collaboration [19].

There are many real-world examples of a single or inter-organizational SOA at work. Amazon uses SOA to create a sales platform with 55 million active customers, and more than one million retail partners worldwide. Up to 2001, Amazon ran a monolithic, very inflexible, and vulnerable to failures Web server application that created the customer and vendor interface, and a catalog. Recently, Amazon’s operation is a collection of hundreds of services delivered by a number of application servers that provide the customer interface, customer service interface, the seller interface, billing and many third-party Web sites that run on Amazon’s SOA platform [5]. A typical inter-organizational SOA coordination is Dollar Rent A Car’s systems using Web services to link its online booking system with Southwest Airline’s Web site. Although both companies’ systems are based on different technology platforms, a person booking a flight on Southwest.com can reserve a car from Dollar without leaving the airline’s Web site. Dollar used Microsoft .NET Web services technology as an intermediary to get Dollar’s reservation system to share data with Southwest’s information systems [11].

Virtually all major IT leaders such as IBM, Microsoft, Oracle, SAP, Sun, HP, and some software vendors specialized in SOA as BEA Systems, TIBCO Software, Software, Sybase, Xcalia, Systinet, Zend Technologies provide tools or entire platforms for building SOA services, integrating software applications and easy of deployment business operations using Web services. Many of above listed companies collaborate their efforts in identifying requirements and designing standards to make data and applications services easier to build and maintain with recent and forthcoming products and to protect the resources and investment. Current publications and discussions in various forums, including the Open SOA Collaboration alliance support the need for an explicit Data Access Service initiative that builds upon SDO (Service Data Objects) and SCA (Service Component Architecture) to standardize important aspects of data services from the consumer’s viewpoint [2].

According to a new Evans Data Corp. study, enterprise adoption of service-oriented architecture is expected to double over the next two years. Evans Data's recently released Corporate Development Issues Survey showed that nearly one-fourth of the enterprise-level developers surveyed said they already have SOA environments in place, and another 28 percent plan to do so within the next 24 months [16]. To facilitate the evolution of applications and make quicker responses to the consumers and businesses needs, the leading companies in SOA structure better the shared IT services by designing horizontal and vertical layers such as: presentation services with profiles, business process and activity services, data and connectivity services, SOA messaging, event processing, management, security and governance. Layering functionality enables IT systems to offer efficiently tailored capabilities to a wide variety of service consumers, to easy adapt to new business conditions by generating an updated services or composing new applications [1].

## 4 The Implication of SOA within UC Models

Business and IT costumers want to achieve greater agility in their business processes and larger variety of service applications, whereas utility computing providers want to reduce costs by consolidating computing power, data storage, information services and network infrastructure. How well do the UC models synthesize with the agility provided by SOA philosophy to enable a continuous optimization of business processes and to satisfy both providers and consumers.

The primary technologies that support utility computing model such as virtualization, SOA and provisioning can mutually interact, be complimentary to one another and became key enablers for flexibility, efficiency and agility of IT utility like services if they are designed and implemented correctly. Dynamic provisioning of the service provider side of the SOA, using virtualization techniques, offers significant gains to utility providers when they build their SOA services on a virtualization platform. There are also benefits from the virtualization of the application or service consumer side based on removing the traditional operating system and allowing more virtual machines to be accommodated on a physical machine. Partial virtualization of the SOA service infrastructure is advisable if the service consumer is new either to virtualization or to SOA technology. Parts of the service may reside on a virtual platform, while other parts of it may reside on a physical one. A J2EE application can communicate with legacy mainframe systems using older protocols, but at the same time present SOA interfaces to its consumers. The J2EE and web services implementation can live on virtual infrastructure while the older legacy systems on its original physical platform. The scheme is followed today with Java/J2EE applications that communicate with legacy systems, and it is equally applicable to the SOA world. When SOA is implemented across the service provider infrastructure, a large collection of services is likely to be present. At a minimum, each SOA service requires a copy of itself to be running on a separate platform, to achieve a level of fail-over and load balancing. When a set of services undergoes an unexpected increase in demand, the whole system must be capable of flexing its processing power to meet that demand, based on the authority and capability of the resource management tools to expand the pool of resources and services [20].

As utility computing becomes more typical, providers and consumers companies have to analyze and reengineer their organization IT resources and processes, and have to develop corresponding changes to IT infrastructure. When implementing SOA, the role of the IT infrastructure changes toward managing the services, which support business processes and therefore, lead to more efficient business results. IT architects with detailed knowledge and understanding to the companies' business needs, processes and expectations have to be involved with more collaborative efforts between previously disconnected professionals like business analysts, infrastructure analysts and application IT analysts to specify and design a new utility computing service oriented infrastructure. Service-Oriented Infrastructure as a shared pool of infrastructure resources that can be dynamically manipulated to align with application requirements, provides more adaptive and with better performance utility computing environment.

As the new ideas for innovation in technology always come from customers and not from technology companies, consumers have a chance to continue to define and refine their requirements so that vendors and service providers would be able to give them what they need. The SOA expanding will stimulate additional performance, flexibility and scalability within services and applications since the immense increase of componentization and standardization into both providers and consumers utility computing infrastructure. When synthesizing SOA philosophy by deploying and realizing business value into utility computing model based on principle of creative frugality: getting the most out of what already exists, rather than replacing technologies that are working effectively, businesses can attain more rapid return on lower investment by acquiring the tools and services that make those technologies more productive and efficient.

## 5 Conclusions

The paper characterizes the utility computing technologies and the paradigm shits consumers, vendors and providers would face when applying partially (selectively) or completely a utility computing model. The role and the advances of SOA approach in this process of utilizing IT services by composing and reusing business-required applications in a utility computing environment has been discussed. Service-oriented computing is a new enormously complex and challenging trend implementing many technologies that must be elaborate in a coherent manner. The framework of SOC might bring more complexity and logical classification of creating composite in-house solutions with external components residing in a virtual utility provider environment.

## References

1. BEA Systems: *BEA's SOA Reference Architecture: A foundation for Business Agility*, BEA Systems, Inc. San Jose, CA 95131, U.S.A. (2008)
2. Carey, M.: S*OA What?* Computer, March 2008, Volume 41, Number 3, IEEE Computer Society, NY 10016, U.S.A. (2008)
3. Carr, N.: *The End of Corporate Computing*. MIT Sloan Management Review, Vol. 46 No. 3, Cambridge, Massachusetts, U.S.A. (2005)
4. David, P., Wright, G.: *General Purpose Technologies and Surges in Productivity: Historical Reflections on the Future of the ICT Revolution*. Oxford University Press for the British Academy (2003)
5. Grey, J.: *Learning from the Amazon Technology Platform*, ACM Queue, No. 4, U.S.A. (2006)
6. Gray, P.: *Manager's Guide to Making Decisions about Information Systems*. John Wiley & Sons, NJ, U.S.A. (2006)
7. Hammond, S.: *Utility Computing: Building the blocks*. ComputerWorld, Hong Kong (2006)
8. Humphreys, J.: *Themis Delivers Policy-Based Automation Across an Application Portfolio*. IDC, MA, U.S.A. (2007)
9. IBM Global Business Services: *Changing the way industries work: The impact of service-oriented architecture*, IBM Global Services, Route 100, Somers, NY 10589, U.S.A. (2006)
10. Ivanov, I.: *Utility Computing: Reality and Beyond*. In: ICE-B'07, International Conference on E-Business (2007)
11. Laudon, K., Laudon, J.: *Management Information Systems: Managing the Digital Firm, (10th edition),* Pearson Prentice Hall, Upper Saddle River, NJ 07458, U.S.A. (2006)
12. Mendoza, A.: Utility Computing: Technologies, standards, and strategies. Artech House, Norwood, MA 02062, U.S.A. (2007)
13. Microsoft Corp.: *SoftGrid® v4: Application Virtualization and Streaming*. U.S.A. (2006)
14. Papazoglou, M. and Ribbers, P.: *e-Business: organizational and technical foundations*, John Wiley and Sons, West Sussex, England (2006)
15. Roberts, J. and Yacono, J.: *Server Virtualization Offers Many Opportunities*. CRN: Iss.1076, NY, U.S.A. (2003)

16. SOA World Magazine: *SOA Adoption to Double in Enterprise* Available: http://soa.sys-con.com/read/358785.htm (May2008)
17. The 451 Group: *Grid Technology User Case Study: JP Morgan Chase*. The 451 Group Report, NY, U.S.A. (2003)
18. Thickens, G.: *Utility Computing: The Next New IT Model*. Available online at: http://www.darwinmag.com/read/040103/utility.html (July 2007)
19. Turban, E., King, D., McKay, J., Marshall, P., Lee, J., and Viehland, D.: *Electronic Commerce 2008: A Managerial Perspective*, Pearson Prentice Hall, Upper Saddle River, NJ 07458, U.S.A. (2008)
20. VMware: *SOA and Virtualization: How Do They Fit Together?* A White Paper from BEA and VMware, Palo Alto, CA 94304, U.S.A. (2007)

# A Comparison of Data and Process Mediation Approaches

Rodrigo Mantovaneli Pessoa[1], Dick Quartel[2] and Marten van Sinderen[1]

[1] University of Twente, 7500 AE Enschede, The Netherlands
[2] Telematica Instituut, 7500 AN Enschede, The Netherlands
{r.mantovanelipessoa, M.J.vanSinderen}@ewi.utwente.nl
{Dick.Quartel}@telin.nl

**Abstract.** In recent years, a huge amount of effort has been invested in the area of service discovery and composition. However, surprisingly little effort is being put into the evaluation of these approaches. The SWS Challenge is an ongoing and continuous experiment in developing a common understanding of various technologies intended to facilitate the automation of mediation, composition, and discovery for Web Services using semantic annotations. The mediation scenario problems concern making a legacy order management system interoperable with external systems that use a simplified version of the RosettaNet PIP3A4 specifications. The participants are supposed to be evaluated with focus on functional coverage. However, it turned out that it is extremely difficult to assess this in an objective manner. In this paper, we describe a framework for comparison of data and process mediation approaches. As a case study, we apply our framework to perform a comparative analysis of four participants from the SWS Challenge.

**Keywords.** Data and Process Mediation, Enterprise Application Integration (EAI), Semantic Web Service Challenge

## 1 Introduction

One of the most engaging promises of Service Oriented Architectures (SOA) is to enable the construction of flexible and loosely coupled business applications, spanning over several networked enterprises capable of interconnecting their applications and share data by combining a set of services. As services mature to suit the basic building blocks of Service Oriented Architectures, the service composition paradigm is becoming one of the main concerns of the application development process. Some already raised questions related to services are: how to specify them in an expressive enough language, how to compose them, how to discover them through the distributed environment, and how to ensure their correctness.

However, the multiplicity and diversity of the proposed approaches attests a lack of consensus on the most appropriate technologies and methodologies to compose services. The Semantic Web Service Challenge is an initiative aiming to develop a common understanding of various technologies intended to facilitate the automation of mediation, discovery and composition for services using semantic annotations. The evaluation process is performed by teams composed of workshop organizers and peer participants with focus on evaluating the functional coverage, i.e. on whether a particular level of the problem could be solved by a particular approach. However it turned out that it is extremely difficult to assess this in an objective manner [1].

Motivated by this fact, we developed a framework for comparison of mediation approaches. The framework is expressed in terms of quantitative and qualitative evaluation points in order to clarify and expose different aspects involved in features supported by a method or tool. As a case study, we applied our framework to perform a comparative evaluation of four participants from the SWS Challenge, around the mediation scenario.

The remaining of this work is structured as follows: Section 2 presents the mediation scenario proposed by the Semantic Web Services Challenge. Section 3 introduces our comparison framework. Section 4 describes four different approaches for data and process mediation. In Section 5, the comparison is conducted and summarized. Finally, Section 6 presents our conclusions and defines some future research directions.

## 2 The Mediation Problem: Purchase Order Scenario

This session describes the static mediation scenario proposed by the Semantic Web Services Challenge. This problem centres around a simple purchase order scenario between two companies: Moon and Blue. The manufacturer Moon has signed an agreement with the company Blue, to exchange purchase order messages in RosettaNet PIP 3A4 format. RosettaNet is an industry-driven standard for B2B integration that represents an agreement on the message exchange patterns, the message content and a secure transportation mechanism among business trading partners in a supply chain network. The Blue’s system has to interact with Moon’s legacy system, also provided as a set of Web services, which however do not use the RosettaNet standard. The objective of the SWS Challenge is to build a system called Mediator, which compensates the differences in communication between the involved parties by solving possible data and behaviour mismatches.

The subsequent levels of the SWS Challenge addresses the mediation problem by asking its participants to, while minimizing direct intervention from programmers, effectively and quickly react to incremental changes of the application requirements built on top of the static scenario. Those solutions that were still able to tackle the problem are then ranked in different levels of adaptability.

### 2.1 The Static Mediation Scenario

The static scenario involves the mediation between two companies, Blue and Moon, within a stable (static) context: the protocols, the messages, and the data formats are known a priori and fixed. In the scenario discussed above, the company Moon uses two back-end (legacy) systems to manage its order processing, namely, a Customer Relation Management System (CRM) and an Order Management System (OM).

As illustrated by Figure 1, the customer Blue sends a RosettaNet order request and expects that, upon the request being submitted, the order will be processed and a purchase order confirmation will be received, acknowledging that the order was received and processed by the company Moon. Messages in RosettaNet PIP 3A4 format enable a buyer to issue a purchase order and to obtain a quick response from the provider that acknowledges which of the purchase order product line items are accepted, rejected, or pending. As mentioned before, the company Moon only offers a set of legacy Web Services that do not fit with the RosettaNet standard. The mediator is in charge of receiving a single RosettaNet message (containing all the order details) from Blue and splitting it to the various messages needed by Moon to create and handle a purchase order. In this way, the mediator will have to orchestrate a sequence of services provided by Moon and translate the set of confirmation messages into a whole RosettaNet Purchase Order Confirmation to be sent back to Blue.

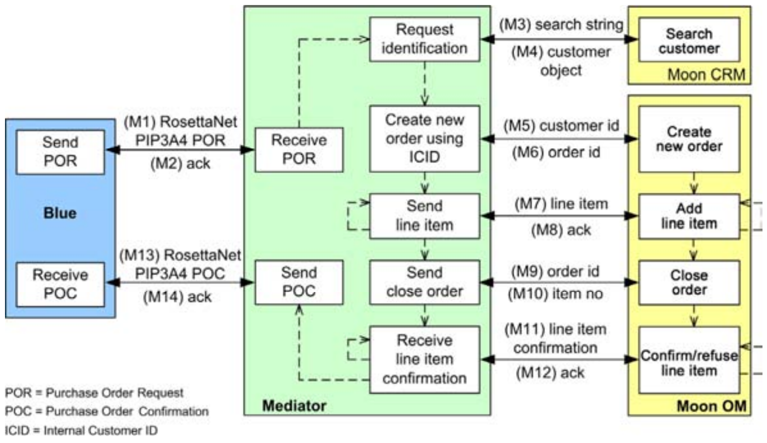


**Fig. 1.** Mediation Scenario Overview.

At first, the Mediator receives a Purchase Order Request message from the customer Blue. The Purchase Order Request message is synchronously confirmed by an Acknowledgement of Receipt message. However, in order to orchestrate Moon to process a RosettaNet purchase order, several steps have to be made.

First, the customer needs to be identified by sending a search string to Moon's CRM system. The internal costumer identification number is obtained by invoking the *searchCustomer* operation. As a next step, the creation of a new order is requested by sending the costumer identification number to Moon's OM system invoking and invoking the *createNewOrder* operation, which returns the id of the newly created order. After a new order is created, Moon's OM system expects all order lines to be added one by one by invoking *addLineItem* operation (possibly for many times). Finally, once all the line items are submitted, Moon OM system is requested to close the order (*closeOrder* operation) and returns the number of items that has been received. Subsequently, Moon's OM system confirms the status of each order line, which is acknowledged synchronously the mediator. After all order lines have been confirmed, a RosettaNet PIP3A4 Purchase Order Confirmation message is sent to Blue and confirmed synchronously by an Acknowledgement of Receipt message.

## 3 The Comparison Framework

This section describes our framework for comparison of mediation approaches. In order to develop our framework, we use the DESMET method [2], a comprehensive methodology for assisting organisations and academic institutions to plan and execute unbiased and reliable evaluation exercises. This method identifies such an evaluation as a qualitative or subjective evaluation and enables the framework to be expressed in terms of a set of common (mandatory and/or desirable) features supported by a method or tool.

Quantitative or objective evaluations are based on identifying the expected benefits and drawbacks of a new method or tool in measurable terms. Qualitative or subjective evaluations assess the appropriateness of a method/tool in terms of the features provided by the method/tool, the characteristics that distinguish this method/tool from others, support offered by the method/tool supplier and its training requirements. This type of analysis is usually based on the identification of the requirements that potential users have for performing a particular task and the mapping of those requirements to features that a method/tool (intend to support that task) should possess. The main activities involved in carrying out a feature analysis are [2]:

1. Select a set of candidate method/tools to evaluate.
2. Decide upon the required properties or features of the item being evaluated.
3. Prioritise those properties or features with respect to the requirements of the method/tool users.
4. Decide the level of confidence that is required in the results and therefore select the level of rigour required of the feature analysis.
5. Agree on a scoring/ranking system that can be applied to all the features.
6. Allocate the responsibilities for carrying out the actual feature evaluation.
7. Carry out the evaluation to determine how well the methods/tools being evaluated meet the criteria that have been set.
8. Analyse and interpret the results.
9. Present the results to the appropriate decision-makers.

As shown in Figure 2, our framework involves both qualitative and quantitative elements, structured into five main features: *data mediation*, *process mediation*, *correctness*, *suitability of design concepts* and *level of effort required to drive changes*. Under *data mediation* and *process mediation* features, we consider both design time and runtime aspects of the mediation task. The first refers to the design support provided by each approach as well as the steps needed to implement each solution, whereas the second refers to characteristics concerning their execution.

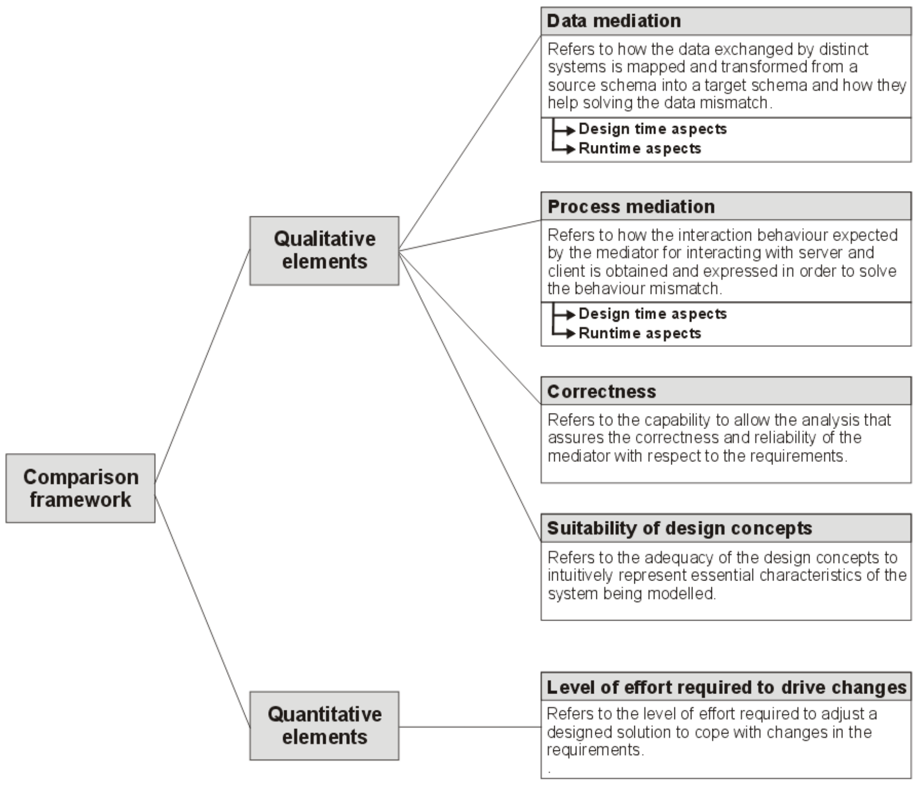

Comparison framework
Qualitative elements
Quantitative elements
Data mediation
Refers to how the data exchanged by distinct systems is mapped and transformed from a source schema into a target schema and how they help solving the data mismatch.
Design time aspects
Runtime aspects
Process mediation
Refers to how the interaction behaviour expected by the mediator for interacting with server and client is obtained and expressed in order to solve the behaviour mismatch.
Design time aspects
Runtime aspects
Correctness
Refers to the capability to allow the analysis that assures the correctness and reliability of the mediator with respect to the requirements.
Suitability of design concepts
Refers to the adequacy of the design concepts to intuitively represent essential characteristics of the system being modelled.
Level of effort required to drive changes
Refers to the level of effort required to adjust a designed solution to cope with changes in the requirements.

**Fig. 2.** The Comparison Framework Elements.

In addition to the separation between quantitative and qualitative evaluations, there is another dimension to an evaluation: the way in which the evaluation is organised. DESMET has identified three rather different ways of organising an evaluation exercise, including: formal experiment (where many subjects are asked to perform a variety of tasks using the different methods/tools under investigation), case study (where each method/tool under investigation is tried out on a real project) or a survey (where subjects that have used a specific method/tool on past are asked to provide information about the method or tool). As mentioned before, as a case study we have adopted the mediation scenario proposed by the SWS Challenge, where different mediation approaches addressing the same real world problem scenario have been peer reviewed and documented.

## 4 Data and Process Mediation Approaches

In this section, we briefly describe different approaches proposed to address the mediation scenario offered by the SWS Challenge. Based on past studies, we have selected four well-documented approaches which have shown some distinctions in their realization.

### 4.1 WSMO, WSML and WSMX

The DERI (Galway and Innsbruck) team based its solution on the Web Service Modelling eXecution environment (WSMX) [7]. WSMX is a reference implementation of the Web Services Modelling Ontology (WSMO) [6] and operates using the Web Services Modelling Language (WSML) [8]. The approach incorporates four core elements that are needed to represent semantic web services and related issues: ontologies, that provide the common terminology used by other WSMO elements, services that are requested, provided, and agreed upon by requesters and providers, goals that represents a desire that a client delegates (which should be solved by services), and mediators, which deal with interoperability problems between different WSMO elements.

During design time, the design and implementation of adapters, creation of WSMO ontologies and services, rules for lifting/lowering, and mapping rules between ontologies are carried out for the RosettaNet, OMS and CRM systems. The run-time phase involves discovery, selection and execution of the appropriate services to mediate the interaction between Blue and Moon systems. The general view of the approach is shown in Figure 3.

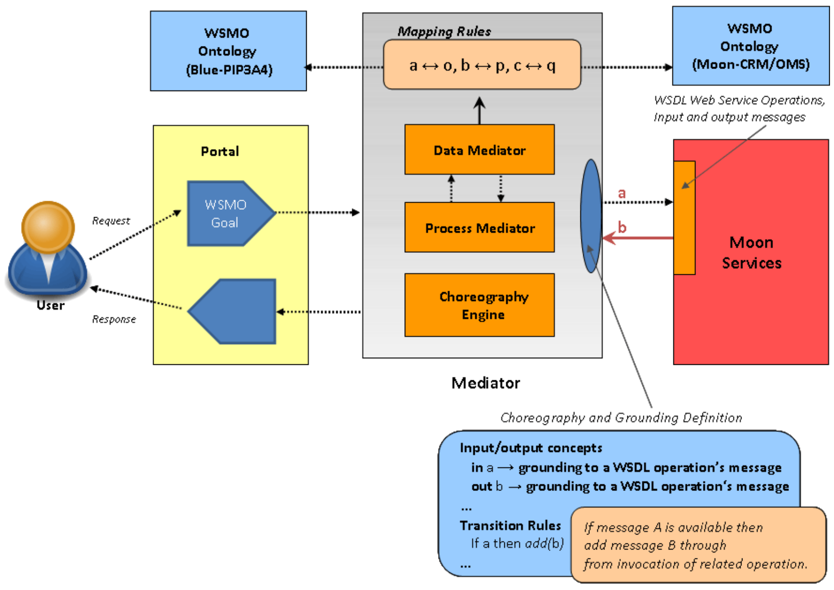

WSMO Ontology (Blue-PIP3A4)
Mapping Rules
a ↔ o, b ↔ p, c ↔ q
WSMO Ontology (Moon-CRM/OMS)
WSDL Web Service Operations, Input and output messages
Portal
WSMO Goal
Request
Response
User
Data Mediator
Process Mediator
Choreography Engine
a
b
Moon Services
Mediator
Choreography and Grounding Definition
Input/output concepts
in a → grounding to a WSDL operation's message
out b → grounding to a WSDL operation's message
...
Transition Rules
If a then add(b)
...
If message A is available then add message B through from invocation of related operation.

**Fig. 3.** General view of the approach.

Initially, ontologies describing the information model used by each involved party are manually designed, after careful analysis of the schemas for the RosettaNet messages and the WSDL service descriptions offered by CRM and OMS systems. In the given scenario, both Blue and Moon use different information models and the data mediation is accomplished through mappings between RosettaNet and CRM/OMS ontologies. In particular, a mapping can specify that classes from two ontologies are equivalent while transformation rules use logical expressions to unambiguously define how the data encapsulated in an instance of one class can be encapsulated in instances of the second class. During run time, if there is a need for data to be mediated, the source instances are provided to the data mediation component, which has the role to derive the target data instances from the source data instances.

In WSMO, requestors of a service express their objectives as goals, which are high level descriptions of concrete tasks. From this point of view, a WSMO goal description consists of a requested capability and requested interfaces. The former shall specify the objective to be achieved in terms of a capability from the client perspective. The latter is intended to specify the communication behaviour for automated Web service usage supported and required by the client. A goal template is a generic objective description that is defined at design time and a goal instance denotes a concrete client request that is created at runtime by instantiating a goal template with concrete values. One advantage of this approach is that the requestor only has to provide a declarative specification of what it wants, and does not need to have a fixed relation with the Web Service or to browse through an UDDI registry for finding Web Services that provide the appropriate capability.

In order for this goal to be accomplished, the requestor has to find an appropriate Web Service which may fulfil the required task. Similar to the way the requestor declares its goal, every Web Service has to declare its capability (that is, what it is able to accomplish) in terms of its own ontology. A WSMO Web service description consists of two central parts. At first, the capability describes the overall functionality provided by a Web service in terms of pre-conditions, assumptions, post-conditions, and effects; these are logical expressions, specified e.g. in WSML. Secondly, the interfaces describe the interaction behaviour supported by a Web service. To cope with impossibility of service requester and provider to communicate with each other due to heterogeneity of their communication protocols, WSMO introduces the mediator concept, which has the task of overcoming the heterogeneity problems, both at data level and at behaviour level.

The WSMX process representation is similar with the WSMO choreography definition, which representation is based on Abstract State Machines (ASM), consisting of states and guarded transitions. A state is described by the WSMO ontology and the guarded transitions (transition rules) are used to express changes of states by means of transition rules. It falls into process execution based on underlying rich knowledge base formalism where an ASM is used to abstractly describe the behaviour of the mediator. In the utilized Abstract State Machines (ASM), the domain ontology constitutes the underlying knowledge representation and transition rules (specified in terms of logic formulas) describe how the state changes when a transition is executed. For the purposes of the SWS Challenge, the provided solution has the assumption that the invocation order is unimportant, but that is not the case: there is an order in which the operations should be correctly invoked.

At this point, both Blue and Moon back-end systems have semantically rich descriptions of the information models and behaviour (choreography) of both systems. This, along with additional mappings between the ontologies of the Blue and Moon systems, allows both choreographies to “connect” at run-time and resolve process interoperability issues (mediate between both choreographies). One of the main advantages of the WSMX-based integration is the strong partner de-coupling. As opposed to traditional centralized solution (when a central workflow would solve this integration problem), this approach enables the automatic adaptation when changes to service descriptions are introduced. In contrast, solutions based on a central workflow would additionally require changes to the workflow type definition.

## 4.2 SWE-ET: Semantic Web Engineering Environment and Tools

The team composed of Politecnico di Milano and CEFRIEL based its solution on the SWE-ET [3] framework. SWE-TE is a framework for designing and developing Semantic Web Service applications, based on existing models for the specification of business processes (such as BPMN [4]) combined with Web engineering models for designing Web applications (such as WebML [5]), with strong emphasis on graphical process modelling.

The approach aims to lead the designer from the process modeling to the running Web application by producing some intermediate artifacts (BPMN models, data models, hypertext models). Such models are enriched by imported ontological descriptions and transformed into a WSMO specification: the ontology is derived from the process model, data model, and hypertext model; the service capability description is derived from the hypertext model; and the choreography information is derived from the process model and the hypertext model. Later, the execution is delegated to a Semantic Execution Environment (e.g. WSMX). Figure 4 provides an overall picture of the approach.

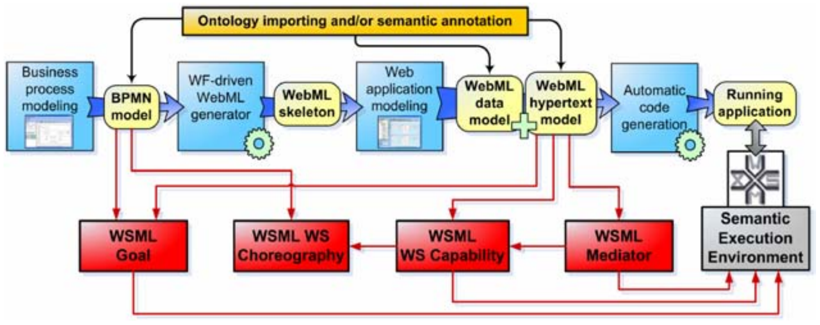


**Fig. 4.** Overall picture of the approach.

The specification of the mediator consists of a set of models: the application data model (an extended Entity-Relationship model), one or more hypertext models (i.e., providing different site views for different types of users), expressing the navigation paths and the page composition of the Web application; and the presentation model, describing the visual aspects of the pages.

Initially, the RosettaNet message schemas and the service descriptions offered by Moon systems were analysed and a corresponding data model was manually obtained from it. The WebML data model is the standard Entity-Relationship (E-R) model and the conversion from RosettaNet messages is handled by Adapter units that use XSLT for transforming messages in an XML format compatible with WebML's internal data format (WSML). In the same way conversion to and from Moon legacy messages are handled by proper XSLT stylesheets that act as templates for SOAP messages and that are then populated by runtime queries.

After modeling the data structures, a high level Business Process Modelling Notation (BPMN) model is created representing the mediator. This model formalizes the orchestration of the Moon Web services and defines states pertaining to the mediation process as by the SWS Challenge specification. The BPMN notation allows one to represent all the basic process concepts such as data and control flow, activity, actor, conditional/split/join gateways, event and exception management, and others. BPMN activities can be grouped into pools, and one pool contains all activities that are to be enacted by a given process participant. The elements of the workflow model (e.g., activity, names, and lanes) are extracted as semantic concepts and used as additional piece of the ontology. If a lane is identified as a mediator at the BPMN level, the basic information about the design of the mediation can be extracted from high-level BPMN description of the interactions (in particular, basic information about possible choreography, interface and parameters of the service).

Then, the BPMN model is used to automatically generate a WebML skeleton that is manually refined. The WebML [5] service model allows one to define different hypertexts (e.g., for different types of users or for different publishing devices), called *site views*. A site view is a graph of *pages*, allowing users from the corresponding group to perform their specific tasks. *Pages* consist of connected *units*, representing publishing of atomic pieces of information, and operations for modifying the underlying data or performing arbitrary business actions. Units are connected by links, to allow navigation, parameter passing, and computation of the hypertext from a unit to another. The WebML conceptual model offers standard workflow units to model control flow and has been extended with Web service units to describe Web services interactions. These units correspond to the WSDL classes of Web service operations, including request-response and one-way operations. Distributed processes can be obtained by combining the workflow units and Web services units. The language is extensible, allowing for the definition of customized operations and units.

Once the business process has been designed, workflow constraints must be turned into navigation constraints among the pages of the activities of the hypertext and into data queries on the workflow metadata for checking the status of the process, thus ensuring that the data shown by the application and user navigation respect the constraints described by the specification.

Then, the WSMO description of the mediator can be derived from the WebML diagrams. This specification can be used to generate a working Web Service providing mediation between Blue and Moon Web Service.

### 4.3 jABC/jETI Framework

The jABC/jETI solution is realized within the jABC framework [9], an environment for model-driven service orchestration based on lightweight process coordination. jABC originated in the context of the verification of distributed systems and use SLGs (Service Logic Graphs) as choreography models, allowing users to easily develop services by composing reusable building blocks into (flow-)graph structures. These basic building blocks are called SIBs (Service Independent Building Block) and the development process is supported by an extensible set of plug-ins that provide additional functionality.

SIBs have one or more edges (branches), which depend on the different outcomes of the execution of the functionality represented by the SIB. Each SLG model can be wrapped into a single coarser-grained SIB, and may be used on another hierarchical level of modelling. Similarly, each SIB can be refined into an own model, showing a more detailed view on the represented feature. The provided model driven design tools allow modelling the mediator in a graphical high level modelling language and supports the derivation of an executable mediator from these models. Figure 5 shows an overview of the described approach.

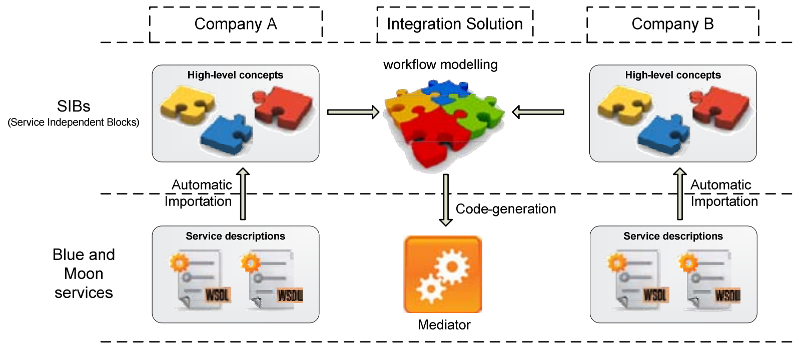


**Fig. 5.** Approach Overview.

Initially, the corresponding SIBs are automatically generated from the WSDL descriptions of the web services provided by the Moon legacy system. At this step, the SIB generator extracts the information about the functions defined in the WSDL service descriptions and creates a SIB for each function. The structure prescribed by the original WSDL service descriptions and RosettaNet Schemas is then mapped into the structure of the SIB parameters, using the pre-existing graphical user interface of the jABC. As a result, the messages are created within the SIBs according to the structure prescribed by the original WSDL descriptions, which is reflected and mapped into the hierarchical parameter structure of the SIBs.

These parameters and the SIB branch labels are visible to the model checker, which allows automatically proving global compliance constraints on the business logic of an SLG. These constraints are expressible in mu-calculus and its derivatives, a family of modal (temporal) logics. Additionally, arbitrary relations between data elements can be provided as local checking expressions, with the expressiveness of Java. This facility allows expressing and checking pre and post conditions.

Next, the mediator is manually modelled as a workflow, by dragging and dropping elements from the palettes of standard and generated SIBs. The modelling activity can then be complemented by analysis, verification and simulation techniques, provided by a set of plug-ins. At this point, the mediation model consists of a structured coordination graph and is interpreted by the tracer plug-in as a flow graph with one or more distinguished start nodes.

To export the mediator as a Web service, the composite and hierarchical SLG of the mediator is first transformed into a single SIB, using the *subgraph* feature of the jABC. This creates a Graph-SIB that represents the corresponding SLG. Its implementation is the argument SLG, executable within the jABC Tracer, the interpreter (or a virtual machine) for SLGs. The tracer is able to execute the mediation model comparable to a standard debugger in *run mode* or *step mode* and using *breakpoints* or *pause* to stop the execution. However, to provide a Web service mediator that is completely independent of the jABC, the code generator plug-in is used to obtain executable source code from the Graph-SIB. This code is then deployed on a server using the AXIS framework, this way making the functionality accessible to other users and generating a WSDL description that contains all the necessary information to access the deployed service as a web service.

## 4.4 COSMO Framework

This approach proposes the use of the COSMO framework [12] for service modelling and refinement in order to raise the level of abstraction at which problems such as mediation and integration of legacy systems are usually solved. In terms of Model Driven Architectures, this means that platform-specific (service) models (PSMs) of Blue and Moon are transformed into platform-independent (service) models (PIMs) by removing all platform-specific details. Next, the approach adds additional semantics to the service PIMs of Blue and Moon in order to make them more precise (e.g. the semantics of service requests and the relations among service operations are explicit modelled). In this way, the solution of the mediation problem is captured in the service PIM of the Mediator. In the final step, a concrete implementation (the mediator PSM) is derived from this PIM by adding technology-specific details. The approach is illustrated in Figure 6.

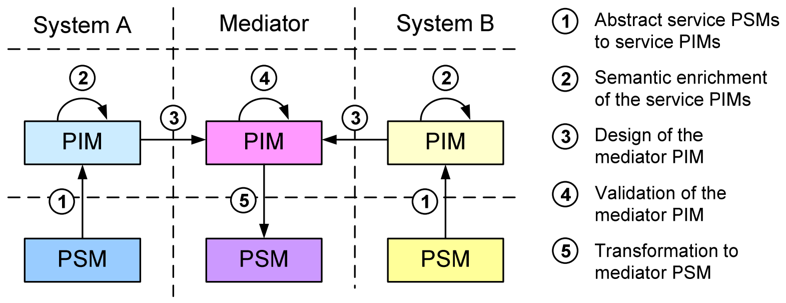

System A
Mediator
System B
PIM
PIM
PIM
PSM
PSM
PSM
1
2
3
4
5
1 Abstract service PSMs to service PIMs
2 Semantic enrichment of the service PIMs
3 Design of the mediator PIM
4 Validation of the mediator PIM
5 Transformation to mediator PSM

**Fig. 6.** General view of the approach.

First, to cope with the problem of data mismatches, the platform-independent information models of the Blue and Moon, hereby referred to as domain-specific ontologies and expressed in OWL, were partially derived using the *types* section of the WSDL descriptions of Moon and Blue systems. The platform-independent behaviour models are partly derived using the *interface* section of the WSDL descriptions of Moon and Blue. These behaviour models are expressed using Interaction System Design Language (ISDL) and the lifting of the interface section of WSDL to ISDL is supported by an integrated editor and simulator for ISDL.

A WSDL *types* section defines only the syntax of the messages that are exchanged between the service provider and its users. Therefore, some further manual work is required to define the semantics of these messages (e.g. hidden assumptions should be made explicit by defining new classes and relations among them). Next, mappings between classes, properties and individuals from Blue's and Moon's domain-specific ontologies are defined.

A WSDL *interface* section defines only its constituent messages and message exchange patterns in a *single* operation. Hence, the complete behaviour model should also define the relationships between the different operations. Since these relationships are not part of the WSDL descriptions they have to be derived from the informal textual descriptions as provided in the mediation scenario. In this way, the integrated behaviour model, describing the possible message exchanges between Blue's and Moon's services, is manually refined from combining concepts provided by the COSMO framework and defining the relationships between their executions.

The core concept underlying the COSMO framework constitutes the *interaction* concept, which represents an activity in which the involved systems produce some common result in cooperation. An interaction is defined by a composition of two or more *interaction contributions*, which represent the participation (or responsibility) of each system involved in the interaction. Consequently, an interaction is considered an atomic activity that either occurs and establishes the same result for all involved systems, or does not occur for any of the systems and therefore does not establish a (partial) result. Additionally, the *action* concept provided by the framework models an activity performed by a single entity and the *causality relations* model how actions and interaction contributions depends on other actions or interactions contributions.

Once the integration solution is specified at the business service layer, it can be early subjected to various analysis and simulation techniques. This is done by applying horizontal transformations to the service model, which are related to transform the service behaviour into a formal specification, which can be then tested and verified to assure the correctness of the derived design with respect to its specification.

After the validation and simulation of the interaction models specified at the business service layer, an IT integration solution can be semi-automatically derived by applying a number of model transformations and refinements. In this step, the behaviour model of the mediator is transformed into a BPEL specification. However, before this mapping can be applied a preparatory step is needed in which the behaviour model of the mediator is annotated with marks and possibly restructured. Marks are used to add implementation details (e.g.: interaction contributions should be marked to indicate whether they have to be mapped onto an invoke, receive or reply activity in BPEL). Furthermore, information about partner links and invoked web services (e.g., namespace URI and endpoint address) may have to be provided.

## 5 Comparison

Table 1 summarizes, according to our framework, the profiles of the proposed solutions, which are commented and described in more detail below:

**Table 1.** Comparison of the described approaches.

| | | WSMO | WebML | jABC | COSMO |
|---|---|---|---|---|---|
| **Data mediation** | **Design time aspects** | Ontologies manually created from analyzing the RosettaNet messages and WSDL service descriptions.<br><br>Ontology to Ontology mappings. | ER-model manually created from analyzing the RosettaNet messages and WSDL service descriptions.<br><br>XML to Ontology mappings. | SIBs and hierarchical parameters automatically generated from WSDL service descriptions.<br><br>XML to SIB parameters mapping. | Ontologies partially generated from RosettaNet messages and WSDL service descriptions.<br><br>Ontology to Ontology mappings |
| | **Runtime aspects** | Mappings execution on the instance level. | Mappings execution on the instance level. | Reflected into the hierarchical parameter structure of the SIBs. | Mappings execution on the instance level. |
| **Process mediation** | **Design time aspects** | Defining services capability, choreography interfaces and goal templates.<br><br>Behaviour modelled as Abstract State Machines by means of transformation rules. | Defining BPMN model, hypertexts and constraints.<br><br>Behaviour specified at a high level of abstraction is transformed into a hypertext model for further manual refinement. | Defining a workflow explicitly describing the behaviour of the mediator.<br><br>Behaviour modelled in terms of control flow graphs based on fork/join parallelism. | Defining a workflow explicitly describing the behaviour of the mediator.<br><br>Behaviour modelled in terms of interactions, operation calls and causality relations. |
| | **Runtime aspects** | Execution based on abstract state machines and transformation rules defined by choreography. | WebML model is transformed into a WSMO specification and execution is delegated to a Semantic Execution Environment (WSMX). | Model-to-code transformation s are defined to generate the implementation code and the execution tree is defined as the unfolding of the marking graph of the mediator. | Simulator tool able to execute the behaviour models. In addition, the Mediator was transformed into a BPEL process and its execution delegated to a BPEL engine. |
| **Behaviour correctness** | | No explicit support. | No explicit support. | Formal verification capability based on temporal logic formulas expressed in mu-calculus. | Formal verification capability based on ISDL techniques. |
| **Suitability of design concepts** | | Appropriate (mediators, goals, services and ontologies). | Sufficient, but not intuitive (pages, units, hypertexts, and links). | Appropriate (Service Independent Building Blocks and hierarchical parameters). | Appropriate (Goals, operations and Interactions). |
| **Level of effort required to drive changes***<br>***Assessed by peer reviews at the SWS workshops.** | | Low<br>(level 3) | Low<br>(levels 3) | Medium<br>(level 2) | Not evaluated |

The profiles presented in Table 1 illustrate that, while the primary aim of the four approaches summarized above is to solve the mediation problem described by the SWS Challenge, their realization differ in several important aspects.

The WSMO approach reflects its four top elements by explicitly modelling goals, mediators, services and ontologies. Ontology-to-ontology mediation is achieved through the design and implementation of adapters specifying mapping rules between ontologies. The approach stresses the importance of the mediators, treated as first class citizens, as the core concepts to describe elements that overcome interoperability problems. Goals are described as requested capability and requested interfaces. From the perspective of a Goal description, the capability describes the functionality that the owner of the Goal wishes to achieve from a Service. Analogously, the capability of a Service describes the functionality offered by that service. The approach focus was on modelling semantically enhancing Web Services description, services requests (expressed as goals) and mediators. The adopted goal-oriented paradigm facilitates the Web Service's discovery by a potential client, the selection of the most appropriate service for a certain task, the actual invocation of a service and the composition of multiple services for accomplishing a common task.

On the other hand, the other approaches focus more on the modelling of the mediator internal logics. The WebML approach starts modelling a BPMN workflow, specified at a high level of abstraction. This model is then transformed into hypertext diagrams, representing the service execution chains, and need to be refined later by the designer. The design concepts provided by the hypertext diagram, originally developed in the context of conceptual modelling of Web pages and applications, were adapted to the mediation purpose and showed to be sufficient to model a mediation solution, but not in an intuitive way. The data mediation is handled by Adapter units that are configured by a proper XSLT stylesheet that transforms messages in an XML format compatible with WebML’s internal ontology format.

The jABC approach automatically imports basic service types (called SIBs, Service-Independent Building Blocks) from the WSDL service descriptions. The designer is then responsible for the specification of the behaviour models, defined as SLGs (Service Logic Graphs), by composing the reusable building blocks into (flow-)graph structures. Behavioural properties of the modelled business logic can be expressed as logic formulas and the provided model, which describes the mediator behaviour, can be analysed in early stages of the design process to check the correctness with respect to its specification. Formal verification capability of the service models is greatly appreciated since it simplifies debugging complex processes directly on the model, possibly reducing development cycle time and increasing robustness of the system. The approach handles data mediation by mapping the structure prescribed by the original WSDL service description into hierarchical SIB parameters (additional semantic properties attached to the SIBs). A derivation of an executable mediator from these models is obtained by applying model-to-code transformations.

Similarly to the WSMO approach, the COSMO approach employs ontologies as the underlying information model. This allows for reasoning to assess whether the relations defined between classes and properties are violated at the instance level or if a common interaction result can be established by matching input and output services parameters. Based on the selected match, the signature for the required data transformation can be obtained automatically. In particular, the approach focuses in applying reasoning techniques to automate parts of the mediator design process. The mediator behaviour is specified as a workflow explicitly modelling interactions between services, operation calls and causality relations between then. Formal verification and analyse of the behaviour models is also supported. The simulator tool is able to execute the behaviour models by performing real web service invocations and incorporating the results that are returned by web services into the simulation. In addition, the Mediator was transformed into a BPEL process and its execution delegated to a BPEL engine.

The level of effort required to adapt each mediator solution to cope with the new changes proposed to the mediator scenario has been assessed by peer reviews at the SWS workshops. For practical reasons, these assessments were adopted and incorporated in our comparison study. There are four possible levels of success that evaluate the transition of the designed solution from one problem level to another. The initial mediation scenario, described in section 2, corresponds to level 0 (static mediation). On top of this static scenario were added various levels, each corresponding to a general kind of problem, and each with sublevels of complexity. In this sense, a higher evaluation success level indicates a better solution to the problem level transition. Since the COSMO team only participated in the first edition of the workshop, their solution has not been assessed by peer review yet.

## 6 Conclusions

In this paper, we have presented a framework for comparison of data and process mediation approaches. The proposed framework establishes a common set of criteria that provide basic guidelines for the evaluation process, enabling a more comprehensive understanding of existing mediation approaches by exploring and making more explicit their possibilities and limitations. In order to assess the features and aspects defined in our framework, the DESMET method for Feature Analysis has been used. This type of analysis identifies an evaluation as a quantitative or qualitative evaluation. In particular, our framework involves both objective and subjective elements and the assessment to which the approaches provide the required features was based on literature review and personal opinion.

As a case study, we applied our framework to perform a comparative analysis of four approaches aimed to solve the mediation problem described by the SWS Challenge. The mediation scenario is pretty close to a real world integration problem involving data and process mediation and has showed to be complex enough to stress the compared solutions. In addition, by applying our framework, we could expose and evidence the advantages and drawbacks of each approach and show that their realization differs in several important aspects.

With our framework, we hope to help the SWS Challenge community by describing and comparing these approaches and providing a comprehensive overview about the underlying concepts, assumptions and promising practices of each approach, including methods, principles and techniques involved in data and process mediation tasks.

## Acknowledgements

This work is part of the Freeband A-MUSE project. Freeband is sponsored by the Dutch government under contract BSIK 03025.

## References

1. Petrie, C., Margaria, T., Käuster, U., Lausen, H., Zaremba, M. (2007): SWS Challenge: status, perspectives and lessons learned so far. In Proceedings of the 9th International Conference on Enterprise Information Systems (ICEIS2007), Special Session on Comparative Evaluation of Semantic Web Service Frameworks, Funchal, Madeira-Portugal.
2. Kitchenham, B. (1996): DESMET: A method for evaluating Software Engineering methods and tools Technical Report TR96-09. Department of Computer Science, University of Keele, Staffordshire.
3. M. Brambilla, I. Celino, S. Ceri, D. Cerizza, E. Della Valle, F. M. Facca (2006): A Software Engineering Approach to Design and Development of Semantic Web Service Applications, In Proceedings of the 5th International Semantic Web Conference (ISWC 2006), Athens, GA, USA, 5-9 November 2006, LNCS 4273, pp. 172-186.
4. White S. A. (2004). Business Process Modeling Notation (BPMN), BPMI.org, http://www.bpmi.org/ bpmi-downloads/BPMN-V1.0.pdf
5. S. Ceri, P. Fraternali, and M. Matera (2002): Conceptual Modeling of Data-Intensive web Applications. IEEE Internet Computing, 6(4).
6. D. Roman, U. Keller, L. Lausen, J. de Bruijn, R. Lara, M. Stollberg, A. Polleres, C. Feier, C. Bussler, and D. Fensel (2005): Web Service Modeling Ontology, Applied Ontologies, vol. 1, pp. 77-106.
7. Mocan, A., Moran, M., Cimpian, E., and Zaremba, M. (2006): Filling the gap - extending service oriented architectures with semantics. In ICEBE, pp. 594-601. IEEE Computer Society."
8. de Bruijn, J. , H. Lausen, A. Polleres, D. Fensel (2006) The Web Service Modeling Language WSML: An Overview. In Proceedings of the 3rd European Semantic Web Conference (ESWC 2006), Budva, Montenegro: Springer, LNCS 4011.
9. Steffen, B., Margaria, T., Nagel, R., Jörges, S., and Kubczak, C. (2006). Model-Driven Development with the jABC. In Proceedings of Haifa Verification Conference, LNCS N.4383. Springer Verlag.
10. Müller-Olm, M., Schmidt, D., and Steffen, B. (1999). Model-checking: A tutorial introduction. In SAS, 6th In: Static Analysis Symposium, LNCS N.1694, pages 330–354. Springer Verlag.
11. Dick A. Quartel , Maarten W. Steen , Stanislav Pokraev , Marten J. Sinderen (2007): COSMO: A conceptual framework for service modelling and refinement, Information Systems Frontiers, v.9 n.2-3, p.225-244.

# Modeling Requirements Elicitation Process for Web Applications

Marian Cristian Mihăescu[1], Cosmin Stoica Spahiu[1], Mihai Mocanu[1] and Bogdan Logofatu[2]

[1]University of Craiova, Faculty of Automatics, Computers and Electronics
Software Engineering Department, Bvd. Decebal, Nr. 107, 200440, Craiova, Dolj, Romania
{mihaescu, stoica_cosmin, mocanu}@software.ucv.ro
[2]University of Bucharest, CREDIS Department
Bd. M. Kogalniceanu 36-46, Sector 5, Bucuresti, Romania
logofatu@credis.ro

**Abstract.** Requirements engineering plays a critical role within a software development process. Studies have revealed a lack of systematic processing of requirements due to high number of activities that need to be accomplished in this phase. There are many reasons for this situation. One of the most difficult tasks is to model requirements elicitation process. This is the first and one of the most important steps from the implications point of view in the nest steps of the development process and in the quality of the obtained software. This paper presents a requirements elicitation technique that has been successfully used in the development process of a web application. The direction of improvements are represented by requirements traceability and their modeling during elicitation phase.

**Keywords.** Requirements elicitation, traceability, modeling, web application.

## 1 Introduction

Ideally, software development, based on any of the well-established life cycle models described in [9] or [10], is linear, starts from scratch, and its phases can be (logically) delimited. No matter the software development model referred, distinct identifiable phases such as, requirements specification and analysis, architectural (overall) design, component design, implementation, component integration, validation and verification, code maintenance and evolution, can be extracted. If and how much are they really separated and easy to reveal, this is a difficult matter related to the specific and goals of the software development process applied.

In practice, software development must deal with the strong tendency to overlap development phases, due perhaps to social issues, and problem domain traditional "ways of doing things", little considered in the past within the discipline of software engineering. The facts are clear: software developers make mistakes, clients change their requirements while the software product is being developed, errors in operating software cannot be avoided a.s.o. The Winburg mini-case study [10] proved all these issues in the most convincing manner. From the implementation of the first version of a software product, continuing to the episode when a fault is found (i.e. due to a slow product operation), and a new design has to be adopted (i.e. using a faster algorithm), or an episode when the requirements change (i.e. because the accuracy has to be increased), the only thing that is “stable” and could be easily noted is a *recurrence of problems*.

Requirements engineering, based on requirements specification, elicitation and analysis, is not only the first, but also one of the most critical, knowledge-intensive activities of software development [1] The most basic question in requirements engineering is how to find out what users really need. Research has shown that in general many large projects fail because of inadequate requirements and specifically that poor execution of elicitation will almost guarantee that the final project is a complete failure. Since project failures are so rampant [2], it is quite likely that improving how the industry performs elicitation could have a dramatic effect on the success record of the industry [3]. Improving requirements elicitation requires us to understand it first. Although many papers have been written to define elicitation, or to prescribe a specific technique to perform during elicitation, nobody has defined yet a unified model of the elicitation process that emphasizes the role of knowledge.

The motivations of this paper can be seen in our effort to reach such a model. We’d like to make the necessary steps in establishing and describing (documenting) the requirements engineering phase, with an emphasis put on requirements elicitation, based on our experience in large software projects. Furthermore, we want to illustrate how we assessed the correctness of all these steps, based on a realistic software product development. In this paper, our specific aim is to discuss all the improvements regarding requirements elicitation that have been tested under real circumstances when developing an e-Learning platform called Tesys [4].

The organization of the rest of the paper is as follows. First, the paper presents an overview of the requirements engineering process (involving requirements specification, elicitation and analysis). Then it presents briefly the Tesys application platform. Next, it discusses the proposed improvements to requirements elicitation, regarding traceability and modeling of requirements in the elicitation process, and the benefits regarding verification and validation The paper concludes with a discussion of the proposed solutions and makes recommendations on how requirements elicitation could proceed.

## 2 Requirements Specification, Elicitation and Analysis

The purpose of the requirements engineering process, which involves requirements definition (specification), elicitation and analysis is to document the requirements for the next phases to be implemented during the software process according to the specific aims of the project. From a social point of view, this should be a collaborative process involving domain expertise from the client and software expertise from the contractor part. The key concept behind this step consists of the separation of application requirements from business/process requirements, which later on permits to link these to design objects. The up-front separation of application from business requirements really helps to clarify and focus on each aspect of the user's requirements, or even helps to determine who should be asking for information about the requirements.

The requirements analysis methodology should cover the entire cycle, from the initial requirements-gathering phase through the separation phase where requirements and non-requirements are set apart. The overall steps required to define an approved set of requirements, and reach closure of the glossary definitions, should be:

1. Collection of 'raw' requirements – This step may start with the domain area experts (in this case the physicians) to write a few pages of text describing the problem to be solved. All relevant sources of possible requirements must be collected. Multiple sources of domain knowledge must be found, to allow the verification of sources. Specialized software tools, such as EasyRM Requirement Management Suite (http://www.easy-rm.com/) or AnalystPro (http://www.analysttool.com/) could be used from this point onwards.
2. Refining of requirements - The objective of this step is to identify the key business concepts, and their properties. Each requirement must be processed in turn for:

– Exclusion for all the following processing steps of any requirements, which are outside the scope of the mission statement,
– Decomposition into sub-requirements, containing only one concept.
– Naming convention – making sure that the name of the requirement reflects the requirement content.
– Redundancies elimination and resolving of duplicate requirements.
– Adjustment of 'priority' (initially all requirements will have the same low priority) and 'status'. Once all requirements have at least the status of 'Approved' then the requirement work has been completed to the point where the requirement set can be passed to the modelers of both teams (client team and implementation team) – possibly to act from here as a 'joint' team, for the creation of the conceptual models.

3. Establishing of reference sources for requirements – Many requirements are directly derived from mission papers, and other conceptual documents. A document manager tool can allow links to be established to these documents.
4. Creation of a document reference (the so-called 'requirements document'), in a bottom-up manner, with a short description giving an overview of what the other documents contain, and a location field used to point to the actual document – this can be filled with a path in the file system, or an URL. As the number of documents may increase, it is essential to classify these in a hierarchical manner, into folders required for specific subject areas.
5. Pass the 'requirements document' to system modelers, to check with the conceptual model they created in parallel with steps 3 and 4.

The end product of this succession of steps should be a requirements document and a correct system conceptual model which should allow the successful development of a software framework and possibly an open source architecture for the software product.

## 3 Tesys Application Platform

An e-Learning platform that represents a collaborative environment for students, professors, secretaries and administrators has been designed and developed. Secretary users manage sections, professors, disciplines and students. The secretaries have also the task to set up the structure of study years for all sections. The main task of a professor is to manage the assigned disciplines. The professor sets up chapters for each assigned discipline by specifying the name and the course document and manages test and exam questions for each chapter. The platform offers students the possibility to download course materials, take tests and exams and communicate with other involved parties like professors and secretaries.

The Tesys platform has initially been designed and implemented only with core functionalities that allowed involved people (learners, course managers, secretaries) to collaborate in good conditions. The requirements engineering followed an ad-hoc process that informally followed the classical life-cycle: elicitation, modeling, analysis, validation, verification and management. The involved parties were represented by three parties: development team, beneficiaries and end-users. Firstly, a prototype that implemented main functionalities has been developed. The requirements were elicited and negotiated between development team and beneficiary. After prototype has been deployed the e-Learning system has been populated with data and users. The beneficiary was the one that kept a close relation with end-users and closely looked the effectiveness of the platform.

The e-learning platform consists of a framework on which a web application may be developed. On server side we choose only open source software that may run on almost all platforms. To achieve this goal Java related technologies were employed.

The model is represented by DBMS (Data Base Management System) that in our case is represented by MySQL [11]. The controller, which represents the business logic of the platform is Java based, being build around Java Servlet Technology [12]. As Servlet container Apache Tomcat 5.0 [13] is used.

This architecture of the platform allows development of the e-learning applica-tion using MVC architecture. The view tier is template based, WebMacro [14] tech-nology being used. WebMacro is also a Java based technology the link between view and controller being done at context level. The separation between business logic and view has great advantages against having them together in the same tier. This decoupling makes development process more productive and safer. One of the biggest advantages of not having business logic and view together is the modularity that avoids problems in application testing and error checking.

In the figure 2 there are presented the main software components from the MVC point of view. MainServlet, Action, Manager, Bean, Helper and all Java classes represent the Controller. The Model is represented by the DBMS itself while the Webmacro templates represent the View. The model is built without any knowledge about views and controllers.

The business logic of the application uses Java classes. As it can be seen in figure 2, there are four levels of dependency between classes. The levels are: servlets, actions, managers and beans. Servlets level has so far two of them: MainServlet and DownloadServlet.

The MainServlet first job first job is to initialize application's parameters. For this purpose the init() method is used. Firstly, there is initialized a pool of database connections. Helper classes like ConnectionPool or ExecuteQuery based on the information from database.properties configuration file conduct this process. In the database configuration file there are set the address of MySQL server and the username and password of MySQL user that is used.

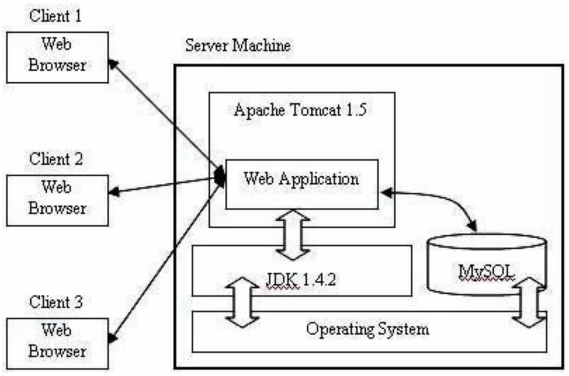


**Fig. 1.** Software architecture of the platform.

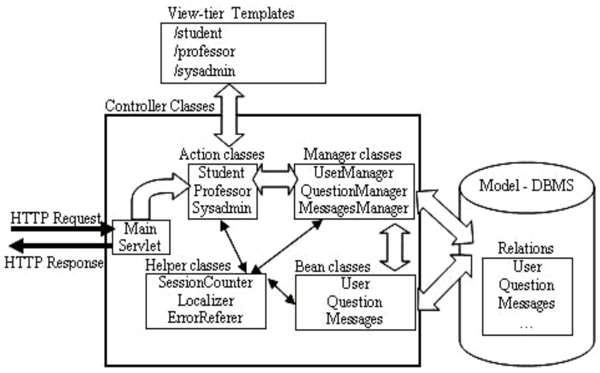


**Fig. 2.** Software components of the application from MVC point of view.

Another important part of software architecture regarding software development process is unit testing. For this purpose JUnit [15] is used. Unit tests are created for running the critical code like creating of a test, computing the result, saving the questions from the test, saving the test result, computing time for test. To accomplish this regressive testing is used. For each chain of actions a scenario is defined. If the computed result matches the expected result it means the test passed. Otherwise, it means something is wrong with the code because it does not behave like it supposed to. Whenever a method is added, test cases are written trying to have a full coverage of the code. There are created batch files that build the code experimentally and continuously and run all the tests. Similarly, a scheduled job runs the nightly build of all the code from the staging area and runs all tests.

The platform is currently in use on Windows 2003 Server machine. This platform has three sections and at each section four disciplines. Twelve professors are defined and more than 650 students. At all disciplines there are edited almost 2500 questions. In the first month of usage almost 500 tests were taken. In the near future, the expected number of students may be close to 1000.

## 4 Improvements in Requirements Elicitation

We present improvements regarding two issues: traceability and modeling of requirements in the elicitation process.

A requirement is defined as an object with his own status and life cycle. The status is determined by the set of values of fields. We define the following set of fields:

**Id** – uniquely identifies the requirement;
**Role** – defines the role to which the requirement addresses;
**Activity** – defines the activity to which the requirement addresses;
**Status** – there were defined three states: INWORK, SOLVED and VERIFIED;
**Solver** – person responsible for implementing the requirement;
**Memo** – text that represents a short summary about the requirement.
**Date** – represents the date when the action has been executed on the requirement
This structure ensures the traceability of the requirement.

- The improvements in requirements elicitation is analyzed from the following points of view [6]:Time and Effort allocation: how requirements elicitation is distributed over time
- Artifacts produced by requirements elicitation: various deliverables like data, effects, results, documents, etc, resulted from requirements elicitation process usage.
- Requirements elicitation Activities: what activities produce artifacts including the requirements specification.
- Disciplines and automation: specify major areas of concern that can influence
- Requirements elicitation, technology and management tools
- Roles: Various roles in RE and differences between them.

The process of modeling the requirements is described in figure1.

The requirements phase has its own life cycle. The specialty literature proves that is difficult to give a general description of requirements activities. In the 80's Krasner identified five phases: need identification and problem analysis; requirements determination; requirements specification; requirements fulfillment; and requirements change management. Another approach presented by Jarke and Pohl in 1994 propose a three-phase cycle of elicitation, expression and validation.

It seems that different approaches use different labels for the requirements activities and this brings about one of the critical problems in requirements engineering: lack of a systematic process. The main problem is that requirements Engineers involves people that communicate with other people. The communication is hard due to the lack of a common scientific language and knowledge. The misunderstandings from this kind of communication are translated directly in wrong application development.

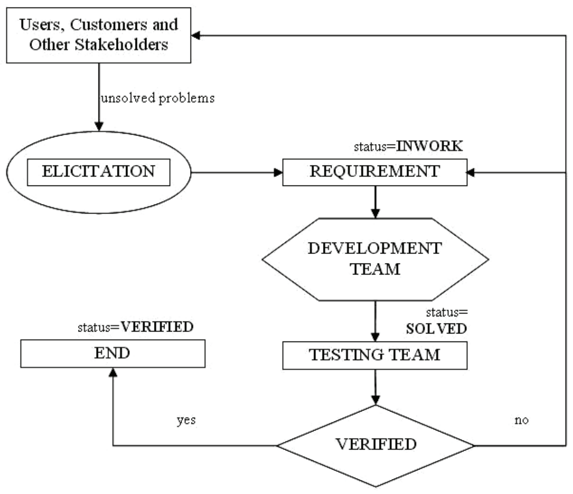


**Fig. 3.** The process of modeling requirements.

The existing problems for requirements elicitation have been grouped into three categories as follows [8]:

- problems of scope (the system edge capabilities are not well defined, unnecessary design information may be given);
- problems of understanding (users don't know exactly what they need and don't know the compter limitations, the users frequently skip "obvious" information, there can be conflicts views from different users
- requirements are often vague and untestable (problems of volatility, requirements evolve over time)

Although this area was not researched enough, there are a series of techniques developed to solve these problems. Traditional methods include brainstorming, interviewing and use of questionnaires[8]. The latests methods for requirements engineering include techniques for information gathering, modeling and representation of information.

Requirements engineering relies fundamentally on verification and validation as a way of achieving quality by getting rid of errors, and as a way of identifying requirements.

One benefit from structuring requirements is the use of automation for verification of requirements. The requirements may be inspected such that verification is performed by using well established checklists. The checklists are applied to the requirements by a well established process.

Modeling requirements in a custom structured form provides the opportunity for analyzing them. Analysis techniques that have been investigated in requirements engineering include requirements animation, automated reasoning, consistency, and a variety of techniques for validation and verification that are further discussed.

Validation is the process of establishing that the requirements and derived structures provide an common and accurate base for involved persons (developers and beneficiaries). Explicitly describing the requirements is a necessary precondition not only for validating requirements, but also for resolving conflicts between developers and beneficiary.

Difficulty of requirements validation comes from many sources. One reason is the problem itself is philosophical in nature. This makes the formalizing process hard to define. On the other hand, there is a big difficulty in reaching agreement among involved persons dew to their conflicting goals. The solution to this problem is requirements negotiation. These will attempt to resolve conflicts between involved parties without necessarily weakening satisfaction of each person's goals.

Structuring requirements brings a big advantage for validation and verification in case of changing requirements. As all successful systems, our e-Learning platform evolves. This means that when a functionality changes because of beneficiary and developer negotiated such a change, this transition needs to be done with minimum of effort. For this, requirements have to be traceable and this feature is accomplished by proposed structuring.

## 5 Conclusions

In this paper, there were presented the main challenges in requirements engineering and especially requirements elicitation. There were presented solutions regarding traceability and modeling of requirements during elicitation phase.

Proposed solutions were tested during Tesys e-Learning platform software development process. It has been also presented the initial requirements engineering process that was used when the prototype has been developed as compared to the improved one.

Proposed solutions come to support a big effort of software globalization development process since the application is rapidly growing in size. More than this, the business logic complexity, degree of heterogeneity among assets is increasing.

Other benefit is that there may be created pools of requirements based on functionality at role level and even with a higher granularity at activity level. This will have a big impact on future decisions regarding what parts of software to be out-sourced in the effort of globalization.

From requirements point of view there were presented three improvements. Finally, there presented the benefits brought by our structuring to verification and validation processes. The proposed structure ensures traceability of requirements, such that as the system evolves the requirements are still properly managed.

## References

1. Gottesdeiner, E.: Requirements by Collaboration, Addison-Wesley, (2002)
2. Standish Group, The Chaos Report, www.standishgroup.com, (1995)
3. Hofmann, H., and F. Lehner: Requirements Engineering as a Success Factor in Software Projects, IEEE Software, 18, 4 (2001)

4. Burdescu, D.D., Mihăescu, M.C.: Tesys: e-Learning Application Built on a Web Platform, Proceedings of International Joint Conference on e-Business and Tele-communications, Setubal, Portugal (2006)
5. Ann M. Hickey, Alan M. Davis, "Requirements Elicitation and Elicitation Technique Selection: A Model for Two Knowledge-Intensive Software Development Processes," *hicss*, p. 96a, 36th Annual Hawaii International Conference on System Sciences (HICSS'03) - Track 3, 2003
6. Bhavani Palyagar, Frank Moisiadis, "Validating Requirements Engineering Process Improvements - A Case Study," *rev*, p. 9, First International Workshop on Requirements Engineering Visualization (REV'06 - RE'06 Workshop), 2006
7. Daniela E. Herlea Damian, "Challenges in Requirements Engineering", Requirements E, Springer, Springer, 2003, vol. 8, no.3, pp. 149-160
8. Michael G. Christel , Kyo C. Kang, "Issues in Requirements Elicitation", Technical Report, 1992
9. Sommerville I., *Software Engineering*, 7th Ed., Pearson –Addison Wesley, 2004
10. Schach S.R., *Object-Oriented and Classical Software Engineering*, 6th Ed., McGraw Hill, 2006
11. Randy Jay Yarger, George Reese, Tim King, "Managing & Using MySQL, Second Edition", O'Reilly, 2002.
12. Jason Hunter, "Java Servlet Programming, 2nd Edition", O'Reilly, 2001.
13. Chanoch Wiggers, "Professional Apache Tomcat", Wiley Publishing, 2003.
14. Faulk, S. "Software Requirements: A Tutorial, Software Engineering", Los Alamitos, CA: IEEE Computer Society Press, 1996.
15. A. Sutcliffe, S. Fickas, and M. M. Sohlberg. PC-RE a method for personal and context requirements engineering with some experience. Req. Eng. J., 11(3):1–17, 2006.

# Dynamic Service Composition: Why, Where and How

Eduardo Silva, Luís Ferreira Pires and Marten van Sinderen

Centre for Telematics and Information Technology, University of Twente
The Netherlands, P.O. Box 217, 7500 AE Enschede
{e.m.g.silva, l.ferreirapires, m.j.vansinderen}@cs.utwente.nl

**Abstract.** We live in a society that is in its nature service-oriented: organizations and individuals get services from others, and provide services to others. This paradigm has been now applied to computer systems with the Service-Oriented Architecture, and it is gaining momentum, mainly motivated by the natural environment provided by the Internet to connect people and businesses. The Service-Oriented Architecture provides an architectural style for the creation, share, composition and execution of networked services. Given the actual dynamic, heterogeneous and distributed nature of computer systems, the composition of services requires mechanisms to support service description, advertisement, discovery, composition, and execution. In this paper we motivate the dynamic composition of networked services, presenting an overview on *why* this area is gaining importance; discussing *where* it has its most promising applications; and finally exposing our initial ideas on *how* dynamic service composition can be realized. To tackle these problems we present a life-cycle for the service composition task, and present our initial framework to support dynamic service composition.

## 1 Introduction

Nowadays we are observing a constant emergence of mobile computing devices, with powerful communication capabilities and increasing processing power. These devices are getting smaller and ubiquitous, and this tendency will continue. Recent studies [1] have concluded that in the upcoming years an increase usage of small devices, referred as *Internet-centric pocketable* devices, will overcome the usage of laptops, mainly for users with high mobility. Such a trend is triggering a change on the way software applications are provided, going from the traditional on-device software applications to Internet-based software applications. This class of Internet-based software applications will take advantage of the high processing power of back-end server systems, providing users with advanced functionality on their pocket computers, offered as services. Therefore, these trends are expected to cause an increase in the usage of Service-Oriented Architecture (SOA) [2]. The acceptance of SOA principles in the design of such distributed software systems will allow companies to sell and buy services based on subscription instead of product licenses. This idea of offering functionality as services (according to the SOA principles) is referred to as *Software as a Service (SaaS)* [3], and allows a client organization or user to use on demand services provided by other organizations or users. Such a change in the way software is provided (as a service) will mainly be possible due the high bandwidth available today, and the way software companies are developing their services, by following open standards, which allows higher interoperability amongst different companies products.

In the context of end-users service provisioning new applications areas are appearing. A clear example is the creation of services on demand, taking into consideration the context (situation) and preferences of the user to adapt the service accordingly [4]. Users' preferences, behaviour, context, etc., vary with the user and his situation, so applications created targeting a large set of users, will not be optimally tailored for all their possible users. Having this idea in mind we claim in this paper that mechanisms for the dynamic composition of services are necessary in order to provide tailored services on demand to service users. We argue that SOA provides the basic principles to support dynamic service composition, but more mechanisms are necessary to improve the collaboration of the different parts of a service-oriented system.

The rest of the paper is organized as follows: Section 2 motivates the dynamic composition of services, why is it required and why is it possible; Section 3 shows some of the possible scenarios for dynamic service composition; Section 4 presents a possible life-cycle for dynamic composition of services; Section 5 presents our initial framework for dynamic service composition; Section 6 presents some related work on dynamic and automatic service composition; and finally Section 7 present some conclusions and discuss challenges to be addressed in the future.

## 2 Motivation

New service applications appear everyday, such as online map services, messaging services, location services, online shopping, etc. This is mainly triggered by the intensive use of the Internet, not only by companies but also by regular end-users, who can create applications and make them available. One of the most popular and successful examples is provided by the open source communities, which are most of the time a group of developers scattered all over the world working remotely (through the Internet) on common projects [5]. The result is a constant increase of available applications, which can be used by different users in different devices. Considering that such applications are made available, for example, on the Internet, new opportunities arise. One of the most interesting opportunities is the creation of new applications out of existing ones, *reuse instead of re-do*. The aim is to allow one to create a new application, in a given programming language, in a given system, and expose it to potential users without requiring them to use exactly the same set of technologies used to develop the application, but instead using the technologies that are more convenient to that application user. However to have such an open architecture all the different parties have to agree on common principles to allow interoperability between the applications. Service-Oriented Architecture (SOA) [2] provides such a set of principles to create distributed computing systems, which supports the creation of loosely coupled applications services in heterogeneous distributed systems.

### 2.1 Service-Oriented Architecture

The Organization for the Advancement of Structured Information Standards (OASIS) defines SOA as [6]:

*A paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations.*

Provided with such principles, developers can create functionality, and make it available to potential users. This functionality is provided as a service to possible users, by defining the service capabilities and how it can be invoked in a service description document. Figure 1 shows the basic concepts behind SOA, such as the different players and interactions required in this architecture.

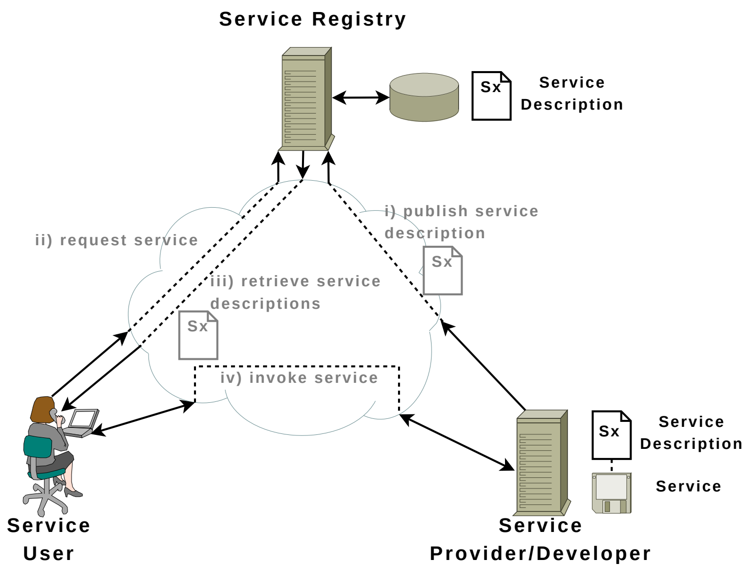


**Fig. 1.** Service-Oriented Architecture concepts and interactions.

SOA is not an implementation technology but a set of principles, that can be implemented in different concrete technologies. One of the most prominent and widely used technologies for implementing the SOA principles is Web services, which is a technology with high industrial acceptance, for which many standards and tools are available. This allows developers or service providers to create and publish services, and allow potential users (Service users) to discover services and possibly invoke them. Some of these standards are Web Services Definition Language (WSDL) [7], Simple Object Access Protocol (SOAP) [8], Universal Description, Discovery and Integration (UDDI) [9], and Business Process Execution Language (WS-BPEL) [10]. They allow one to describe services, exchange messages, publish/discover service descriptions and compose services, respectively. More standards have been developed, which aim at addressing all the additional issues concerning Web services systems.

### 2.2 Why Dynamic Composition

Traditionally service developers make application services from scratch, triggered by a specific request from the service users or by the identification of some potential (business) opportunity. This approach is time consuming, and leads, many times, to the duplication of already existing functionality. Following the SOA principles, service developers can instead create compositions of existing services to fulfill some given user needs. Nowadays there are plenty of tools to support developers in the task of service composition, and these tools tend to facilitate (ease and optimize) this task, enabling the re-use of existing services. However, the current approach consists of the creation of static service compositions, with fixed service end-points, targeting a specific group of service users. We argue that more dynamic composition mechanisms have to be developed to allow the creation of more personalized, adaptable and context sensitive services.

Assuming that different services are available that can be discovered and composed, we claim that more dynamic mechanisms can be used to achieve *on demand* service composition, given a specific user service request. This is the essence of dynamic service composition: *perform the composition of existing services on demand to match specific user requirements and preferences*. In this paper we motivate dynamic service composition based on a specific user service request, so taking the user request, context and preferences into account in the service composition process. Dynamic service composition may also address, such as, for example, the adaptation of a service composition in case a service component is unavailable, implying that an alternative service is discovered to replace the unavailable one, however this is not the focus of this paper. In [11] other research challenges have been identified in the area of service composition, however is clear that much focus is given to the creation of more dynamic mechanisms for service composition.

To achieve dynamic service composition, frameworks to coordinate all the phases of the service composition life-cycle are required. If such frameworks are available, users will be able to develop more personalized services, according to their needs.

## 3 Scenarios for Dynamic Service Composition

The most natural scenarios for service composition are *Internet-based*. This is mainly justified by the big number of applications that are available, which can be exposed as services (e.g., web services). Considering that the providers of these services publish their descriptions in a service registry, other users or service developers can discover and make use of these services. For example, if there are services that provide lists of hotels and lists of taxi companies given a location, a client user may on-demand create a new service that allows to book a hotel and given the location of the hotel book a taxi to take him from the airport to the hotel. Another clear example concerns inter-organisational (business) computing. If different organisations provide specialized services to each other, each organisation can focus on their own expertise and simply outsource some services by reusing other organisation's services to achieve the functionality they require. This inter-organisational cooperation allow partners to reduce the cost and possibly optimize their products, since they can focus on the problem to be solved, avoiding to tackle all the less important issues that are required to solve it. The main issue in this situation, most of the time, is not the service composition process at the technical level, but the contractual issues. This is in our opinion one of the greatest barriers to the actual usage of service-oriented architectures in inter-organizational systems.

Another interesting scenario is concerned with *mobile computing*, in which mobile devices are provided with some functionality, but rely on back-end systems to perform the most complex computations and provide the necessary services. In [12] these services are described as *Field Web services*. The idea of this scenario is to provide the *field* mobile user device with the necessary functionality to interact with the back-end systems, and perform all the more complex computing tasks on the back-end systems. This is an emerging idea, and is gaining a lot of attention from different parties, especially from telecom service providers. If service users are provided with basic frameworks that allow them to discover and compose available services *anywhere and anytime* according to their context and preferences, companies may create a great source of revenue by providing such personalised services. This hybrid system (mobile clients and back-end server system) has a lot of potential applications, and fits the current trend of moving user client to mobile platforms. Another advantage that can be foreseen in applying SOA principles in mobile computing environments is that SOA provides a natural environment for task distribution, which allows one to save battery life of mobile devices. This issue is a very well known problem in mobile computing areas, since it is often a bottleneck for the usage of mobile devices. In [13] several ideas and challenges to the application SOA principles in mobile environments are discussed, as well as how more mature SOA principles applied to wired environments can be adapted to mobile environments.

These examples differ in their nature and application areas. The *Internet-based* scenario seems to be the most natural and also the most suitable scenario for service composition at the moment. However, the *Mobile computing* scenario, due to the intrinsic mobility of the users, provide interesting application opportunities to be explored. We expect that both scenarios will increase the usage of service-oriented architecture techniques, mainly triggered by the flexibility provided by such architectural approach. Dynamic service composition techniques will allow one to address the personalization, context and preferences of the users in any of these scenarios. This implies that efforts have to be made to allow the dynamic composition of services.

## 4 Service Composition Life-cycle

To present our framework for dynamic service composition we first introduce the notion of *service composition life-cycle* for dynamic service composition. Figure 2 shows the different phases and the different perspectives of the service composition life-cycle.

In the context of service-oriented systems different perspectives (or parties) have to be considered in the service composition creation and execution life-cycle. We admit that there are two main perspectives in this life-cycle: the *Service user* and the *Service developer/provider* perspectives. Other authors distinguish the service developer from the service provider. However, to simplify the discussion, we assume that the service

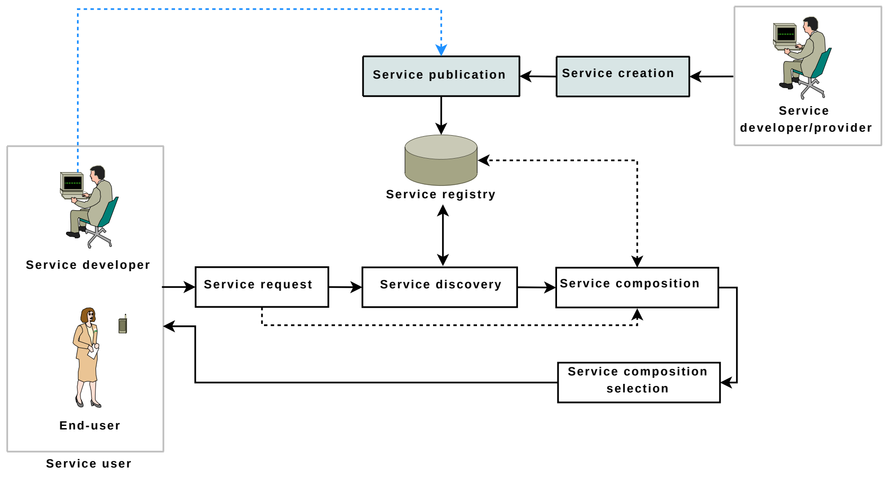


**Fig. 2.** Service composition life-cycle.

developer and the service provider are the same entity in our life-cycle. The service user can be an end-user, a person without much technical knowledge, or can also be a service developer, who makes use of services that are possibly provided by other service developers/providers to create new value-added services.

From the perspective of the service developer/provider, two main phases can be identified: *service creation* and *service publication*. The service creation phase basically focuses on the creation of the application service. An application service may be a new application created from scratch, or may consist of wrapped legacy or existing applications exposed as a service with a well-defined interface. The service creation can alternatively consist on the construction of a new service composition, meaning that a service developer/provider simply re-uses existing services to compose a new value-added service. After the creation of a new service, a service descriptions for this service should be published in a service registry. The publication phase consists on the publication of the service description document, so the service can be later discovered by possible service users.

From the perspective of the service user, several phases can be identified: *service request* specification; *service discovery*; *service composition*; *service composition selection*. The service request specification consists of the definition of the desired service properties and goals. This information is used to perform service discovery, and to drive the service composition and selection processes. Two main approaches to service discovery can be considered: discover all the relevant services for the composition, based on the service request; or iteratively discover the required services during the service composition process. A combination of these two approaches can also be considered, in which all the relevant services are considered first and then extra services are discovered on-demand at composition time, in case the set of discovered services is not sufficient to complete the service composition process. Independently of how the service discovery phase is implemented, it is always made based on information specified in the service request. The next phase is the actual service composition, where an algorithm for the creation of a service composition plan is used to match the user service request for a composition. Given the set of discovered services, different alternative service composition plans may be generated. In this case, the next phase consists on the selection of a service composition, again based on properties of the user service request such as, for example, cost, reliability and response time, and his context and preferences. Additionally, and not stated in the figure, for the end-user case a service deployment phase must also exist, so the service can be deployed to be ready for execution. In Figure 2 there is another possible phase in the perspective of the service user, mainly the service developer - the *Service publication*. This phase consists on the publication of a service that is created dynamically for a given service user. This motivates the use of dynamic service composition mechanisms to support not only end-users, but also service developers, who can then publish the generated service compositions.

In this service composition life-cycle we ignored several issues, such as for example service binding, service deployment. We intentionally ignore these details, leaving them open to be addressed in the concrete frameworks for dynamic composition of services, since these operations can be specified in different phases of the life-cycle.

## 5 Framework for Dynamic Service Composition

Figure 3 presents our initial framework for dynamic service composition, following the life-cycle presented in the Section 4.

Figure 3, shows that our framework makes uses of ontologies for service creation and description, service request description and also for service discovery and the construction of service compositions. In computer science, an ontology consists of a formal specification of a conceptualization of a given domain. This formalization allows

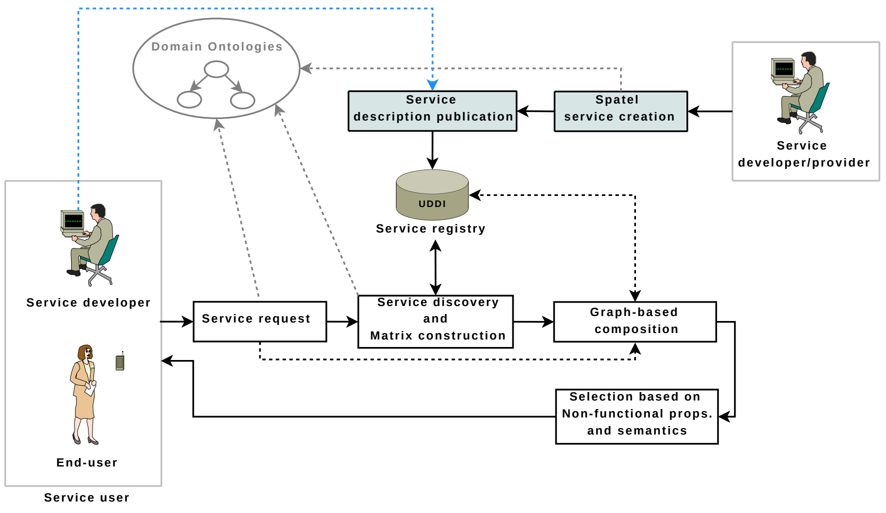

Domain Ontologies
Service description publication
Spatel service creation
Service developer/provider
UDDI
Service registry
Service developer
Service request
Service discovery and Matrix construction
Graph-based composition
Selection based on Non-functional props. and semantics
End-user
Service user

**Fig. 3.** Framework for dynamic service composition.

the description of a domain at a semantic level. This semantic level description enables automatic reasoning, i.e., without human intervention. In our framework we make use of this possibility to perform the service composition based on a service request and the service description of the existing services, both concepts described in common ontologies.

Our framework is being implemented in the context of the SPICE (Service Platform for Innovative Communication Environments) project [14]. In the SPICE project a language called Spatel [15] has been defined to support the creation, composition and execution of services and service compositions. Another property of this language is that it allows semantic annotations to be associated to service operations and properties. Provided with the Spatel language, a service developer can create new services, and semantically annotate them according to ontologies of the service domain. In our framework this is done by the *Spatel service creation* module. Another language could be used that support semantically annotated services, such as for example SAWSDL [16]. After the service is created, such annotations can be used in the *Service description publication* module, to publish the necessary information to enable service discovery. These modules reflect the life-cycle service developer/provider perspective in the framework.

From the perspective of the service user, the first step we consider in the framework is the definition of the *Service request*. The service request has to express the goal(s) of the service user for his service, so that the necessary discovery and composition of existing services can take place. We define a service request, in an XML-based format, as follows:

```
<ServiceRequest>
    <Inputs>..</Inputs>
    <Outputs>..</Outputs>
    <Preconditions>..</Preconditions>
    <Effects>..<Effects/>
    <Goals>..</Goals>
    <Non-functional>..</Non-functional>
    <Ontologies>..</Ontologies>
</ServiceRequest>
```

At the moment we are experimenting with simple stateless services, not taking into account complex service behaviors. The service request above has seven different types of annotations. The service *Inputs*, *Outputs*, *Preconditions* and *Effects* (IOPEs) refer to specific input, output, precondition and effect parameters that the service composition has to contain and satisfy. The *Goals* annotations describe specific goals that the service composition has to fulfill, such as, for example: translate, bookTicket, findDoctor. The *Non-functional* properties specify some additional requirements that the service composition has to fulfill, such as, for example: maximum cost of the service composition and minimum level of security. The *Ontologies* lists the ontologies used to specify the service request properties. This means that each property has to be specified following a defined concept in a valid domain ontology. An example of a service request is the following:

```
<ServiceRequest>
    <Inputs>
        <"LanguageOnt#Language" name="srcLang">
        <"LanguageOnt#English" name="trgtLang">
        <"LanguageOnt#Text" name="txtToTrans">
        <"TelecomOnt#PhoneNum" name="destNumber">
    </Inputs>
    <Outputs>
        <"TelecomOnt#AckSMS" name="AcknowledgmentSMS">
    </Outputs>
    <Preconditions/><Effects/>
    <Goals>
        <"GoalOnt#translate">
        <"GoalOnt#sendSMS">
    </Goals>
    <Non-functional>
        <"NFPOnt#Cost" value=0,50 EUR>
    </Non-functional>
    <Ontologies>
        <"GoalOnt" "TelecomOnt" "NFPOnt" "LanguageOnt">
    </Ontologies>
</ServiceRequest>
```

This service request denotes a service that performs the translation of a text from a given language to English, and sends the result by SMS to a specific telephone number. Furthermore the service should not cost more that 0,50 EUR. This service request specifies the necessary inputs for the service, i.e., the text to be translated, the source and target languages for the translation, and the telephone number to which the message has to be sent. The output of the service request is a simple acknowledgment that the SMS message has been received. The goals are to translate and send an SMS message. Finally, the ontologies used to specify the annotations are also listed in the service request.

Once the service request is available the *Service discovery and matrix construction* module can be used to perform the discovery of the necessary services and organize them into a matrix that facilitates the construction of the actual service compositions by the *Graph-based composition* module. We perform service discovery based on service goals. For example, in the service request above two goals are specified: $GoalOnt\#translate$ and $GoalOnt\#sendSMS$. Based on these goals the service discovery module queries the service registry (an UDDI-based registry, extended with semantic support) for services with goal annotations semantically related with the service request goals. After retrieving all the matching services, they are organized in a *Causal Link Matrix (CLM$^+$)* [17], which is a formalism that allows the representation of all the possible semantic links between the discovered services. By semantic links we mean the connection between services inputs and outputs, which are described with semantic annotations using common ontologies, to allow their composition and interoperability.

Once the CLM$^+$ is created, the *Graph-based composition* module can perform the necessary service composition. The service composition algorithm consists of a graph-based algorithm that uses the service request specification to drive the composition process. It starts from the specified service request outputs and composes services backwards until all the requested service inputs are matched and all the requested goals are resolved. At each iteration, the composition algorithm checks whether the requested non-functional properties are met by the service composition; if these are not met the composition is discarded.

At the end of the process several service compositions may be generated. To help select a particular composition we rank the generated compositions according the specified non-functional properties and the services semantic links. This is an important issue, since if the service user is an end-user without any technological knowledge, he expects to obtain a running service. This implies that a particular composition has to be selected if alternative service compositions are possible. In the future we also intend to take the user's context into account in the selection of the most appropriate service.

We refer to [18] for a discussion in the CLM$^+$ creation, the graph-based composition algorithm and the proposed ranking algorithm for service composition selection.

## 6 Related Work

The area service composition is a very active area of research nowadays. Different aspects of service composition are being studied. However the integration of the different parts of the process of service composition, from the service request to the actual runnable service composition, using dynamic and automatic mechanisms is still not addressed by many.

[19] address the problem of interleaving web service discovery and composition, considering only simple workflows where web services have one input and one output parameter. In this case the web service composition plan is restricted to a sequence of limited web services corresponding to a linear workflow of web services. In our framework we propose a formalism to support the composition of services with multiple inputs and outputs, and also address the other phases of the life-cycle of the service composition process.

In [20] an algorithm for automatic composition of services is presented. The service composition is considered as a directed graph, where nodes are linked by the semantic matching compatibility ($Exact$, $Subsume$, $PlugIn$, $Disjoint$) between input and output parameters. Based on this graph, the shortest sequence of web services from the initial requirements to the goal can be determined. This approach compute the best composition according to the semantic similarity of output and input pa rameters of web services, but it does not consider any non-functional properties of the composition services. We consider this to be a very pertinent point to take into account, since the selection of the most suitable service compositions are many times based on such properties (e.g.: cost, security, etc.).

In [21] a semi-automatic composition process is proposed to perform the composition of web services. This approach supports the system user on the selection of web services for each activity in the composition, and to create flow specifications to link them. The discovery process consists on finding matching services, meaning web services that provide outputs that can be fed as input on the services that exist in the service composition. After selecting all the services, the system generates a composite process in DAML-S [22]. The composition is executed by calling each service separately, and passing the results between services according to the flow specifications. This process grant a higher control over the composition process, which is sometimes desirable for service developers. However, and since the composition process is semi-automatic, end-users without technical knowledge can't usually make use of this approach. Our framework deals with the composition process in a more abstract and automated way, which allow its usage by both service developers and end-users.

## 7 Conclusions and Future Work

In this paper we motivate dynamic service composition. We claim that given the current trends on computer and communication systems an increase usage of distributed application services is expected. This implies that new mechanisms and architectures are required to support such systems, and also to provide users with tools to use these new application services. The main architectural principles to support these ideas can be found in the Service-Oriented Architecture (SOA).

To motivate the creation of mechanisms for dynamic service composition we provide two potential scenarios suitable for service composition: the *Internet-based* scenario, where several services can be published or advertised, and one can make use of them to compose new application services, reusing the existing services; and the *Mobile computing* scenario, which has a lot of potential with the emergence of mobile devices and communications. In the later scenario mechanisms are required to support mobile users, providing them with minimal functionality at the mobile terminal, and performing the complex tasks at the back-end server systems.

We conclude by providing some initial ideas on how to tackle the problem of dynamic composition of services. We discuss a dynamic service composition life-cycle, showing the phases, and perspectives that are necessary to support the process of dynamic composition of services. Based on this we present our initial framework for dynamic composition of services, from the service user request to the actual service composition.

In the future we plan to explore further our ideas and improve the proposed framework towards a generic framework to support dynamic service composition using different technologies in different application scenarios. The following research challenges have been identified:

1. The service request module has to accept user service requests in an abstract form, to support not so technical skilled users. It has also to collect context information and other user preferences, to be used in the composition process.
2. At the moment we perform service discovery based on the specified goals on the service request, but it is clear that other services may be needed at composition time to complete a service composition. Given this, mechanisms to make service discovery at composition time have also to be considered in the framework.
3. The use of ontologies is clearly required to allow such a dynamic mechanism for service composition. Nevertheless how and where these ontologies are defined is still fuzzy. This is an issue that may have very interesting research challenges.
4. The proposed framework is being prototyped, following the proposed modular architecture. The aim is to provide a very modular architecture so one can easily

extend it and support other service description languages, and service composition languages. We plan to evaluate the prototype in a specific scenario in the e-health domain, specifying for this a library of services and also the necessary ontologies to describe the domain.

## Acknowledgements

This work is supported by the European IST SPICE project (IST-027617) and the Dutch Freeband A-MUSE project (BSIK 03025).

## References

1. Gartner: Gartner highlights key predictions for it organisations and users in 2008 and beyond. http://gartner.com/it/page.jsp?id=593207 (January 2008)
2. Erl, T.: Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall (2005)
3. O'Reilly, T.: The open source paradigm shift. In: Perspectives on Free and Open Source Software, The MIT Press (July 2005) 461 – 481
4. Jorstad, I., van Thanh, D.: Service personalisation in mobile heterogeneous environments. In: Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services, IEEE Computer Society (February 2006) 70 – 75
5. Raymond, E.S.: The Cathedral and the Bazaar. O'Reilly & Associates, Inc., Sebastopol, CA, USA (1999)
6. MacKenzie, C.M., Laskey, K., McCabe, F., Brown, P.F., Metz, R.: Reference model for service oriented architecture 1.0. Technical report, OASIS (October 2006)
7. Chinnici, R., Moreau, J.J., Ryman, A., Weerawarana, S.: Web services description language (wsdl) version 2.0. http://www.w3.org/TR/wsdl20/ (June 2007)
8. Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J.J., Nielsen, H.F., Karmarkar, A., Lafon, Y.: Simple object access protocol (soap) version 1.2. http://www.w3.org/TR/soap12-part1/ (April 2007)
9. Clement, L., von Riegen, A.H., Rogers, T.: Universal description discovery and integration (uddi) version 3.0. http://uddi.org/pubs/uddi_v3.htm (October 2004)
10. Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I., Weerawarana, S.: Business process execution language for web services, version 1.1 (May 2003)
11. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F.: Service-oriented computing: State of the art and research challenges. IEEE Computer **40**(11) (2007) 38 – 45
12. Papazoglou, M.P.: Web Services: Principles and Technology. Prentice Hall (2007)
13. Sen, R., Handorean, R., Roman, G.C., Gill, C.: Service Oriented Computing Imperatives in Ad Hoc Wireless Settings. In: Service-Oriented Software System Engineering: Challenges And Practices. Idea Group Publishing (2005) 247 – 269
14. Cordier, C., Carrez, F., van Kranenburg, H., Licciardi, C., van der Meer, J., Spedalieri, A., Rouzic, J.P.L.: Addressing the challenges of beyond 3G service delivery: the SPICE platform. In: 6th International Workshop on Applications and Services in Wireless Networks. (May 2006)
15. Almeida, J.P., Baravaglio, A., Belaunde, M., Falcarin, P., Kovacs, E.: Service creation in the spice service platform. In: Wireless World Research Forum meeting on "Serving and Managing users in a heterogeneous environment". (November 2006)

16. Sivashanmugam, K., Verma, K., Sheth, A., Miller, J.: Adding semantics to web services standards. In: 1st International Conference on Web Services. (2003) 395–401
17. Lécué, F., Léger, A.: A formal model for semantic web service composition. In: International Semantic Web Conference. LNCS, vol. 4273 (2006) 385–398
18. Lécué, F., Silva, E., Pires, L.F.: A framework for dynamic web services composition. In: 2nd ECOWS Workshop on Emerging Web Services Technology, CEUR Workshop Proceedings (November 2007)
19. Lassila, O., Dixit, S.: Interleaving discovery and composition for simple workfows. In: First International Semantic Web Services Symposium. (2004)
20. Zhang, R., Arpinar, I.B., Aleman-Meza, B.: Automatic composition of semantic web services. In: 1st International Conference on Web Services. (2003) 38–41
21. Sirin, E., Hendler, J.A., Parsia, B.: Semi-automatic composition of web services using semantic descriptions. In: 1st Workshop on Web Services: Modeling, Architecture and Infrastructure. (2003) 17–24
22. Burstein, M.H., Hobbs, J.R., Lassila, O., Martin, D.L., McDermott, D.V., McIlraith, S.A., Narayanan, S., Paolucci, M., Payne, T.R., Sycara, K.P.: Daml-s: Web service description for the semantic web. In: International Semantic Web Conference. (2002) 348–363

# Author Index

Proceedings of the Second International Workshop on
Enterprise Systems and Technology - I-WEST
ISBN: 978-989-8111-50-0