Enterprise Interoperability
Enterprise Interoperability

New Challenges and Approaches
Over the past 20 years, the business world has been changing significantly, and concepts of cross-enterprise collaboration have been promoted to the center of enterprise strategy. For some organizations, doing business globally has become critical to their survival; others discover new opportunities by focusing their business on a local setting. Not only large organizations set up cooperation agreements with other enterprises, but also SMEs are combining forces to compete jointly in the market. Nowadays, the competitiveness of an enterprise is largely determined by its ability to seamlessly interoperate with others.

Interoperability in the context of enterprises and enterprise applications can be defined as the ability of a system or a product to work seamlessly with other systems or products without requiring special effort from the customer or user. The possibility to interact and exchange information internally and with external organisations is a key issue in the enterprise sector. It is fundamental in order to produce goods and services quickly, at lower cost, while maintaining higher levels of quality and customisation. Interoperability is considered to be achieved if the interaction can, at least, take place at the three levels: data, applications and business enterprise. Through the architecture of the enterprise model and taking into account the semantics issues. It is not only a problem of software and IT technologies. It implies support of communication and transactions between different organisations that must be based on shared process models and business references.

This book establishes the proceedings of the Second International Conference on Interoperability for Enterprise Software and Applications (I-ESA 06), which

These two programmes were the results of several initiatives launched in 2000 by the European Commission, where an expert group had been set up to identify challenges and approaches relating to the development of interoperability of enterprise applications in Europe, and to make propositions to the E.C. to launch projects to sustainably advance enterprise interoperability. In 2002 a Thematic Network IDEAS (Interoperability Development for Enterprise Applications and Software – Roadmaps), coordinated by University Bordeaux 1, was launched, the project manager being Guy Doumeingts. IDEAS produced an influential Roadmap on Enterprise Interoperability. INTEROP, ATHENA and other projects in the FP6 were built on the content of the roadmap.

The originality of the approach proposed by IDEAS was to find solutions to problems of Enterprise Interoperability by combining three main research domains: Enterprise Modelling (EM) dealing with the representation of the inter-networked organisation to establish interoperability requirements and to support the implementation of solutions; Architecture & Platform (A&P) defining the implementation solutions to achieve interoperability; Ontologies (ON) addressing the semantics necessary to assure interoperability.

I-ESA’06 is the successor and was built on the success of the INTEROP-ESA conference 2005, held in Geneva on February 2005. The objective of I-ESA is to bring together the world's leading researchers and practitioners in the area of Enterprise Interoperability and e-Business and to enable a fruitful exchange of the participants’ vision, research and experiences.

I-ESA’06 was held from March 22nd to 24th (colocated with eGovInterop), 2006. 20th and 21st were devoted to the pre-conference. 47 regular papers were presented in 16 sessions and 29 posters in five sessions. These 24 sessions were held in three parallel tracks.

The programme proposed several key notes presented by high-level renowned experts from industry and academia:

- Dr. Gerald Santucci, European Commission, EU,
- Dr. Steve McGibbon, Microsoft, UK,
- Dr. Michael Brodie, Verizon Information Technology, US,
- Dr. Mark Schenecker, SAP, US,
- Ludger Garg, Siemens Transportation Systems, DE,
- Man-Sze Li, IC Focus, GB and Prof. Jean-Paul Bourrières, Univ. Bordeaux 1, FR,
- Hubert Tardieu, Atos Origin, FR.

During the 20th and 21st, the pre-conference was held featuring:
• a full-day doctoral symposium containing 15 papers,
• three tutorials on model-driven interoperability with SOA and agents, Use of enterprise models to master collaboration challenge; and AKME – Active modelling knowledge of enterprises.
• three full-day scientific workshops:
  - IFAC TC5.3 workshop Enterprise Integration, Interoperability and Networking (EI2N'2006),
  - Web Services and Interoperability (WSI'2006),
• two advanced full-day workshops:
  - Interoperability challenges for SMEs: this event co-sponsored by INTEROP NoE and ATHENA IP concerned a very important topic: the debate were of high quality,
  - Enterprise Interoperability Roadmap, organised, by the European Commission. This session was well attended by more than 80 participants. The subject was the elaboration of the Roadmap in Enterprise Interoperability for the future. The result of this debate with the contribution of Europe, North-America, Japan, China and Australia representatives will serve as a base for the next 7th Research and Development Framework Programme of the European Commission in this domain.
• the 15th anniversary celebration of China/Europe collaboration in the area Integration and Interoperability.

Having attracted 250 attendees, the I-ESA’06 conference demonstrated the importance and prominent role of Enterprise Interoperability for the competitiveness of enterprises. The 250 participants proved an increasing interest for this domain.

The next conference, I-ESA’07 will be held in March 26th-30th in Funchal, Madeira Island, Portugal, organised by the Institute UNINOVA, http://www.i-esaa.org/i-esaa2007.

We believe that this thoroughly prepared volume is of particular value to all readers interested in key topics and most recent developments in the very exciting field of enterprise interoperability research.

Bordeaux,  
Clausthal,  
Nancy,  
October 2006

Guy Doumeingts  
Jörg P. Müller  
Gérard Morel  
Bruno Vallespir
Acknowledgements

The Conference Chairs and the Organizers would like to thank the members of the INTEROP NoE and ATHENA IP for their support and their contribution to the establishment of the 2nd International Conference on Interoperability for Enterprise Software and Applications (I-ESA’06) conference, held in Bordeaux, France, March 20-24, 2006.

We want also to express our gratitude to the European Commission, the Conseil Regional of Aquitaine, the University Bordeaux 1, ENSEIRB, CNRS for their encouragement and their active support through a grant.

The collaboration of IFIP, IFAC, ADEISO, GDR MACS of the CNRS, I3 and the e-GOV project was also highly appreciated.

We are deeply thankful to the local organisation committee, members of the LAPS/GRAI University Bordeaux 1 and ADERA for their excellent work for the preparation and management of the conference.

Finally and most importantly, we want to thank all the authors, invited speakers, reviewers and participants of the conference that made this I-ESA 06 conference and the resulting book a reality and a success.
Contents

Part I Service-oriented Interoperability Approaches

Development of Dynamic Composed Services Based on the Context
F.J. Nieto, L. Bastida, M. Escalante and A. Gortazar ............................................... 3

Semantics of Interoperable and Outsourced Information Systems
H. Balsters and G.B. Huitema .................................................................................. 13

A Platform Independent Model for Service Oriented Architectures
G. Benguria, X. Larrucea, B. Elvesæter, T. Neple, A. Beardsmore
and M. Friess ........................................................................................................ 23

Part II Enterprise Interoperability Architecture

Formalizing Analysis of Enterprise Architecture
P. Johnson, L. Nordström and R. Lagerström ......................................................... 35

To Adapt or Not to Adapt, That is the Question: Towards a Framework to Analyze
Scenarios for Aligning Legacy and Business Components
W.-J. van den Heuvel ............................................................................................ 45

Value object analysis and the transformation from value model to process model
H. Weigand, P. Johannesson, B. Andersson, M. Bergholtz, A. Edirisuriya and T.
Ilayperuma ............................................................................................................ 55

Activity Diagram Based Process Family Architectures for Enterprise Application
Families
A. Schnieders and M. Weske ................................................................................ 67
Part III Model-driven Approaches to Interoperability

An Integrated Model-driven Service Engineering Environment
J.P.A. Almeida, M.-E. Iacob, H. Jonkers, M. Lankhorst and D. van Leeuwen

UML for Enterprise Modelling: basis for a Model-Driven Approach
R. Grangel, J.-P. Bourey, R. Chalmeta and M. Bigand

Solving Problems in the Parameterisation of ERPs Using a Model-Driven Approach
R. Grangel, J.-P. Bourey, and A.J. Berre

A Decentralized Broker Architecture for Collaborative Business Process Modelling and Enactment
B. Bauer, J.P. Müller and S. Roser

Part IV Methods, Models, Languages and Tools for Enterprise Interoperability

Discretization of Continuous Features by Using a Kernel
L. González, F. Velasco, F.J. Cuberos, J.A. Ortega and C. Angulo

Designing and Implementing Cross-Organizational Business Processes - Description and Application of a Modelling Framework
U. Greiner, S. Lippe, T. Kahl, J. Ziemann and F.-W. Jäkel

Quality Criteria for Enterprise Modelling in the Context of Networked Enterprises
T. Knothe and R. Jochem

The UEML Approach to Modelling Construct Description
A.L. Opdahl

A Roadmap for UEML
A.L. Opdahl and G. Berio

An Interoperable Platform to Implement Collaborative Forecasting in OEM Supply Chains
R. Poler, A. Ortiz, F.C. Lario and M. Alba

D. Vanderhaeghen, D. Werth, T. Kahl and P. Loos
Achieving Enterprise Model Interoperability Applying a Common Enterprise Metamodel
J. Ziemann, O. Ohren, F.-W. Jaekel, T. Kahl and T. Knothe................................. 199

Interoperability Characterization Using Enterprise Modeling and Graph Representation
S. Blanc, Y. Ducq and B. Vallespir........................................................................ 209

Part V Semantics and Ontology-based Interoperability

Semantic Service Modeling: Enabling System Interoperability
S. Pokraev, D. Quartel, M.W.A. Steen and M. Reichert ...................................... 221

Supporting Scientific Collaboration in a Network of Excellence Through a Semantically Indexed Knowledge Map
P. Velardi, A. Cucchiarelli and M. Petit............................................................... 231

Mapping XML Schema to OWL
N. Ancic, N. Ivezic and Z. Marjanovice............................................................ 243

Extending OWL-S to Solve Enterprise Application Integration Issues
S. Izza, L. Vincent, P. Burlat, P. Lebrun and H. Solignac ............................... 253

Practical Issues in Ontology Modeling: The Case of Defence Conceptual Modeling Framework-Ontology
A. De Nicola, V. Kabilan, M. Missikoff and V. Mojtahed..................................... 265

Digital Resource Discovery: Semantic Annotation and Matching Techniques
D. Bianchini, S. Castano, F. D’Antonio, V. De Antonellis, M. Harzallah, J. David, M. Missikoff and S. Montanelli................................. 275

A Model for Assessing the Impact of Enterprise Application Interoperability in the Typical European Enterprise
Y. Charalabidis, D. Askounis and G. Gionis........................................................ 287

Ontology-based Transformations for Achieving Interoperability in AmI
S. Erofeev and X. Larrucea.................................................................................. 297

Part VI Interoperability of Decision Models

A Decentralized Approach for Inter-Enterprise Business Process Collaboration
X. Qiao, J. Wei and T. Huang................................................................................ 309
Towards a Conceptualization of Decisional Interoperability
N. Daclin, D. Chen and B. Vallespir ................................................................. 321

Interoperability and Synergism of Decision, Information and Flexibility to Improve Performances of Enterprise Systems: KM Implications
S. Wadhwa, A. Saxena and Y. Ducq ................................................................. 331

Part VII Inter-organisational Interoperability

Service Typing in Collaborative Systems
T. Ruokolainen and L. Kutvonen ................................................................. 343

Decentralized Metadata Development for Open B2B Electronic Business
F. van Blommestein ...................................................................................... 355

The TrustCoM Approach to Enforcing Agreements between Interoperating Enterprises

Organisational Inter-operability: Towards Enterprise Urbanism
F. Biennier and H. Mathieu.................................................................................. 377

An Integrated Approach for Organizational Data Interoperability
J. Sampson, C. Veres and M. Lanzenberger...................................................... 387

Managing the Lifecycle of Cross-organizational Collaborative Business Processes
P. Walter, D. Werth and P. Loos .......................................................................... 397

Interoperability through Model-based Generation: The Case of the Collaborative Information System (CIS)

A Natural Basis for Interoperability
N. Rossiter, M. Heather and D. Nelson............................................................... 417

Designing a Modular Infrastructure for Exploratory Integration of Interoperability Approaches
N. Mehandjiev, P. Grefen, I.D. Stalker, R. Eshuis, K. Fessl and G. Weichart ..... 427
Part VIII Interoperability of Manufacturing Enterprise Applications

Supply Chain Management System and Interoperability through EAI Platform
L. Zhou, X. Xu, T. Huang and S. Deng ................................................................. 441

Enhancing Interoperability of Manufacturing Software Units Using Capability Profiling
W. Yu, M. Matsuda and Q. Wang ........................................................................ 451

Towards a Product Oriented Process Modelling for Enterprise Applications Synchronisation and Interoperability
S. Baìna, H. Panetto and G. Morel ...................................................................... 461

Part IX Business Models Interoperability

Developing Interoperable Business Processes Using Web Services and Policies
Z. Maamar, D. Benslimane, G.K. Mostéfaoui, S. Sattanathan and C. Ghedira ... 475

Talea: An Extensible Framework for E-Business Integration
G. Levi, A Vagliengo and A. Goy ........................................................................ 489

A Complexity Based Approach to Collaborations in the Tool and Die Industry
G. Schuh, A. Sauer and S. Döring ....................................................................... 499

Modelling Inter-organizational Workflow Security in a Peer-to-Peer Environment
A. Nowak, M. Hafner, M. Breu and R. Breu ..................................................... 509

Enterprise Software with Half-Duplex Interoperations
M. Johnson ......................................................................................................... 521

e-Proc a TO BE Scenario for Business Interoperability

Part X Standards for Interoperability

ICT Standards Development – Finding the Best Platform
K. Jakobs ......................................................................................................... 543

Experts on Causes of Incompatibility between Standard-Compliant Products
T. Egyedi ........................................................................................................... 553

Achieving Influence on Standardisation Bodies
E. Söderström .................................................................................................. 565
Interoperability in Healthcare: Standards for the French Project of a National Personal Medical Card

P. Lagouarde, B. Le Blanc and B. Vallespir ........................................................ 575

Index of Contributors ........................................................................................... 585
Part I

Service-oriented Interoperability Approaches
Development of Dynamic Composed Services Based on the Context

Francisco Javier Nieto, European Software Institute, Parque Tecnologico de Zamudio, E-48170 Zamudio-Bizkaia, Spain, francisco.nieto@esi.es
Leire Bastida, European Software Institute, Parque Tecnologico de Zamudio, E-48170 Zamudio-Bizkaia, Spain, leire.bastida@esi.es
Marisa Escalante, European Software Institute, Parque Tecnologico de Zamudio, E-48170 Zamudio-Bizkaia, Spain, marisa.escalante@esi.es
Alayn Gortazar, European Software Institute, Parque Tecnologico de Zamudio, E-48170 Zamudio-Bizkaia, Spain, alayn.gortazar@esi.es

Keywords: Dynamic composition, Context-aware, Services, Methodology

1 Introduction

One organisation interested in working in the field of services composition should answer the following question: which are the main steps in order to create a composed service, either statically or dynamically? In order to answer this question is important to have in mind the framework process [1] in which the composition development is included. This framework process has three main process areas:

- Service development/engineering process area focussing on best practices in the context of developing ‘atomic services’[1]. The primary purpose of this process area is to develop, test and deliver atomic services.
- Service Acquisition/provisioning process area addresses processes and activities related to the service marketplace and the role of service broker.
- Service centric system engineering area. This is the most important area for the purpose of this paper. Service centric system engineering process area includes all the functionality related to the analysis, design, development, deployment and execution of a software system based on services. The component parts involved in this process are similar to those found in

1 Atomic Service is a single service not formed by other services.
traditional software engineering, but concentrated on the development of service-oriented architectures for the system being developed.

This paper is structured as follows. First, it provides a brief introduction about existing approaches for dynamic composition. Second, a proposal for a process to develop static composition is explained. Finally, the paper contains a detailed description of the process for developing composed services based on variability, as well as an explanation and an example of how to model the variability.

2 Approaches for Development of Dynamic Composed Services

In static service compositions, the activities, services and workflows are statically defined at design time. This means, after analysing the requirements, the composition with fixed elements is developed. It is not possible to change the integrated services during the execution of the composition, even when there are errors.

In a dynamic service composition, some approaches have been identified to allow certain dynamicity in the service composition, enabling the business process\(^2\) to adapt to the context and manage the recovery actions when errors are detected. The objectives are to achieve high flexible service compositions, and to improve the reliability of the composition.

The different approaches identified can be used together to give the composition the maximum flexibility, because they have impact at different levels of the service compositions. It is necessary to define specific tasks and processes to fulfil the requirements of the approaches at design time as well as at runtime. These approaches are the following ones:

- **Re-Binding.** This approach enables to replace a service with other, which will be similar and will cover the same functionality. At design time, some realisation alternatives for the overall functionality of the composition are identified. If the default service has to be replaced due to an error or a QoS decrease, the re-binding process will select a new one from the alternatives list, which could be enlarged by discovering new services at runtime, and will modify the composition by binding the new selected service.

- **Variability Points.** In this approach, the parts of the composition where there will be different possible alternatives in order to execute an abstract activity\(^3\), are identified. These alternatives can be identified at design time, but it is not possible to know which one will be executed at runtime. Depending on the context when the composition is requested and the conditions are defined for each variability point, one option will be chosen and executed. The variances may be defined as different available features or as service realisation alternatives at runtime.

---

\(^2\) Business process is understood as the composition flow including additional aspects such as transactions, security issues.

\(^3\) Abstract activity is an activity that not necessarily has a concrete implementation.
• **Re-Planning.** The approach is used when a service participating in the composition fails, and there is no possibility to replace it with other atomic service. In this case, the composition needs to be modified in such a way that the lost functionality can be recovered. This functionality may be split (so it will be covered by various services) or may be regrouped with other part of the composition (a service that is working properly to perform part of the functionality of the composition could be used to perform also the lost functionality).

3 **Static Composition Framework**

A static service composition is a composed service workflow, defined manually and a priori at design time. As consequence, after the service is executed and running, no changes are possible. In this way it is possible to compose really complex services, as interactions between used services are well known and controlled. But this static condition means a lack of adaptability to different users or environments, as it requires knowing very well the consumed services and their interactions, so if adaptability or dynamic compositions are mandatory factors, some actions should be taken during the static design to allow a dynamic service composition at run time. [2]

![Static Composition Framework](image)

**Fig. 1. Static Composition Framework**

In Figure 1, the static service composition methodology is split into five processes: Enterprise Architecture Engineering, Service Specification Architecture, Design Time Service Composition, Composition Realisation Architecture, and Validation process. These processes are composed by several tasks, which are explained more in depth in [2]; including for each step a figure showing these main tasks, using the SPEM notation [6].
3.1 Enterprise Architecture Engineering Process

The purpose of this process is to identify, define and maintain domain or organisation wide architecture models, and assess their corresponding benefits and drawbacks to fulfil system requirements.

To achieve this objective, once the system requirements and the available candidate services are provided by a previous service discovery mechanism, it is important to gather information about previous projects to analyse similar solutions.

Different architectures that may fit the needed composition might be defined or extracted from those solutions. Once some candidate architectures are selected, the advantages and disadvantages of each one should be analysed to facilitate the decision about which architecture will be used for the composition.

The final result of this process will be a list of the available Architectural Options and Patterns, ready to be used as a skeleton for the service composition structure. This information may be enhanced with descriptions of the different alternatives, instructions on how to use them, results of a feasibility analysis and so on. To easy the future choice of an appropriate architecture, some Architectural Policies will also be obtained.

3.2 Service Specification Architecture Process

After obtaining the suitable architecture for the composition, the services that fulfil this architecture should be found or, at least, defined. This process defines how to specify the services that are needed to cover all the functionalities of the system, specifying abstract service specifications for the composition that will only describe their functionalities without any specific implementation.

These service specifications will be used afterwards to find concrete services that can fulfil the requirements. If some services which are necessary to fulfil the requirements can not be found, new services might be developed.

As a result of this process, a Service Specification Model will be obtained; specifying the interface of each identified abstract service, as well as its description and requirements. It could also be possible to specify some mappings with service implementations that cover the functionalities of the identified abstract services.

3.3 Design Time Service Composition Process

In this process, a service composition will be specified taking as inputs the Service Specification Model and the previously described Architectural Policies.

For this purpose, once the abstract services have been specified, the architecture that best fulfils the composition requirements will be selected, taking into consideration the business strategic perspectives. Then, the abstract services that fulfil the architectural requirements must be organized, obtaining an abstract service composition.

Finally, it will be necessary to describe how to fulfil other composition relevant aspects, such as error handling, security or transactions management.
At the end of this process, a document with the modelled abstract representation of the composition will be obtained. This document is called Service Composition Specification, and it also includes more information about the resulting composition, such as workflow, transactions, security issues, etc…

3.4 Composition Realisation Architecture Process

After obtaining the Service Composition Specification, it is necessary to select the final services that fulfil the abstract composition. In this process, candidate services and possible alternatives will be chosen using previously described service specifications. These services can be discovered or developed, depending on the possible candidates and the system requirements, and they will be mapped with the abstract composition specification. The goal is to obtain the final business process that will fulfil the functionality expected from the system during its execution.

The main result of this process will be an Executable Script written in a concrete implementation language for the composition, i.e. WS-BPEL [5], which will include all the issues described in the Service Composition Specification. Furthermore, a Service Deployment Architecture document will also be produced. This document will explain how services will interact with each other and which Quality of Service (QoS) levels will be covered.

3.5 Validation Process

As a final step, the composed service will be validated to assure that selected services provide the appropriate functionalities with the optimal QoS. For this purpose, and to guarantee that no violations or deadlocks are possible, the workflow should be iterated. To assure that the obtained results are always correct, and that the system maintains the required stability, the system should be overloaded during this process.

Finally, the updated versions of the documents produced on the previous process will be obtained; i.e. a Verified Executable Script and a Verified Service Deployment Architecture.

4 Dynamic Composition Framework based on Variability

Software variability has become an important topic in software architectures and product family approaches [3] to derive different product configurations through the use of variation points to describe those parts of a software system that may vary. This approach has been applied to the development of service compositions since different configurations could be possible in a service composition depending on different factors. The main idea of variability for the business process workflow is to introduce one more abstraction layer in the business process. In this way, developers include tasks at design time to define explicitly the variability that can be concreted in later phases and implemented in very different ways.

The application of variability needs to be supported during the design of a service composition, because introducing variability once the business process has
been developed and implemented is very difficult. Maintaining the coherence of
the composition and the fulfilment of the requirements is not a trivial matter. This
happens because adding variability to the composition might affect many tasks as
well as global policies like those for security, transactions and QoS.

To add this abstraction level to the business process, we have proposed to carry
out two extra processes during the development of service composition (Variation
points management and Variation points realisation), which will help developers to
select the composition parts in which to introduce dynamicity using variability, as
well as how to implement and define possible alternatives for each variation point.

These processes are expected to be used in any development process,
improving any methodology in such a way that can guide developers to introduce
some dynamic parts in their compositions at design time. In the figure above, we
show how these processes can be fitted in the static composition framework.

4.1 Variation Points Management Process

As explained before, some parts of the composition might change at run-time. The
purpose of this process is to describe at design time which parts of the composition
might change at runtime and define a set of requirements to perform this variability
and maintain the coherence with the business process goal.

At this point, a Service Composition Specification has been modelled in the
previous step, so the basic abstract business process is defined. This specification
has to be analysed in order to get which parts of the composition might have a
different behaviour at runtime depending on context information. The selected
architecture, the composition requirements and the abstract services descriptions
identified are some of the composition aspects that will determine where the
variable parts might be.

Those variable parts will be defined as variation points and the next step will
consist on describing each one. This is, defining the requirements and the criteria
(functional, non-functional, dependencies…) to be fulfilled in order to guarantee that the new composition will be coherent and compatible with the initial requirements and goal. It is important to put special attention in aspects like transactions, security and QoS policies which are sensitive to changes.

The final result of this process will be an enhanced and improved Service Composition Specification, which will now also include variation point specifications to be instantiated at runtime.

4.2 Variation Points Realisation Process

Until this process, there is not enough information to instantiate variable points. This process guides developers through the process of gathering that information indicating which services are candidates for each variation point and how those services fulfil the requirements of the variation point, as well as defining some rules to indicate when to use one service or another.

Before starting this process, some candidate services should be discovered, and the composition specification should be completed as a service deployment architecture. With this information, composition developers will define the candidate services for each variation point, mapping the requirements of the variation points and the abstract service descriptions. A ranking may be created to facilitate the selection of services and their alternatives.

At the same time, the variation points policies have to be described. To perform this activity, developers will define a set of rules or requirements to guide the execution engine in the services selection task in such a way that can fit better the executed composition with the user requirements and needs. Defining which context information is needed is critical, so the execution engine will know which data must be gathered.

From this process, an enhanced Service Deployment Architecture and an enhanced Execution Script will be obtained. The Execution Script will contain specific language elements to indicate variabilities and policies to manage them, so the execution engine can interpret them correctly. In this way, the execution engine will be ready to instantiate those variation points at runtime before starting the execution of the composition.

5 Modelling Dynamic Composition based on Variability

To model the variability identified in a dynamic composition, it is necessary to define the information required to describe variation points in composite services and to configure them at different points in the lifecycle (e.g. design time or runtime). Then, this information model should be attached to a composition language, such as WS-BPEL [5]. However, the details about how to attach these variability concepts to a composition language are still under our research.

This section provides a definition of the variability concepts used in composition language and also a conceptual model that relates all these concepts. The figure 3 gathers all these variability concepts and their relationships.
A variation point (VP) identifies a point in a composition context which varies according to the value of a decision taken using context information. Each variation is defined with a set of information such as:

- A Name identifying the variation point with a unique identifier within the composition.
- A Description describing the variation point, giving the necessary plain text information to support the process of decision making.
- A ValueType specifying the data type of the variation value.
- A DefaultValue a pre-set value, indicating the value automatically applied to the variation point if the decision is not taken explicitly.
- The Validity, indicating if the variation point has to be taken during composition execution. This attribute is true by default.

A variant is one of the different alternatives in a variation point. Only one variant is active in the system at any given moment. Each variant is defined with a set of information such as:

- A Name identifying the variant with a unique identifier within its owner.
- A Description describing the variant, giving the necessary text plain information about the alternatives gathered in that variant.
- A Value indicating the value associated to that variant. This means, the variant will be the alternative chosen in case that value coincides with the decision taken.
- An Implementation indicating the candidate service attached to the variant. This candidate service could be a single concrete service to an abstract service composition or several abstract/concrete services to replace a part of the composition.

![Variability Conceptual Model](image)

**Fig. 3. Variability Conceptual Model**
Example of Usage

This section provides an example of the usage of variability concepts explained in the previous section. The example shows how a specific configuration impacts on the final composition. Keep in mind that this example only manages simple variability and disregard dependencies that could appear between variation points.

Suppose that we want to use an IP voice service called *kRecall* that, after checking the user identity through an external service denominated *kLogin*, provides different alternative ways to phone, using different telecommunication providers. The *kRecall* service provides two alternative telecommunication providers, ProviderA and ProviderB. If a mobile phone user wants to phone to a ProviderB user, the best choice would be to use the same telecom provider as the target mobile user, this means, Provider B service is preferred.

To model this in the composition, we have to add a variation point for *kRecall* depending on the telecom provider of the target mobile phone and define the associated variants. The best variant is chosen on the basis of the target mobile phone. This decision is taken based on the value given to the variation point associated at runtime.

For example, suppose that Mary uses the *kRecall* service to get her through John and ProviderB is the telecommunication provider that supplies phone services to John. Then, the composition flow of *kRecall* service, after configuring the variation points identified, would be denoted by the following figure:

![Diagram showing the example of variability choices based on telecommunication providers.](image)

Fig. 4. Example of Variability Choices based on Telecommunication Providers

If the target phone provider is ‘ProviderA’, the first alternative or variant will be chosen. If the target phone provider is ‘ProviderB’, then the second variant will be chosen.

Now, the *kRecall* service is configured using the user’s context. Depending on a specific configuration (in this case, the type of telecom provider for target phone), the variation point is instantiated and final the composition flow is generated and specifically adapted to the context.
6 Conclusions and Future Work

Existing services composition approaches focus mainly on static composition based on service description and orchestration standards. Adding variability to a business process is a good solution to get some level of dynamicity.

The static framework presented in the paper can be used as a guide to develop any service composition. Following all the steps, a robust and complete composition could be obtained with a good quality level.

But there are many domains where dynamic compositions offer many advantages, so the paper shows how to apply a product family engineering approach as a feasible solution for developing composed services, as they share some commonalities regarding their variable behaviour in some circumstances. The advantage of this approach is that provides the appropriate functionality with the optimal quality depending on the context in the execution time.

This approach can be considered as a dynamic re-composition since there is already a service composition defined. To perform a real dynamic composition, we need to create a service composition from scratch and in real time. This kind of approach is known by on-the-fly composition, which needs only a few input parameters provided by the user, like a composition goal, to develop low-complex compositions. This approach will be researched in the context of future European projects.

The variability conceptual model included in section 5 describes a first approach. This model should be also improved to express dependencies between variation points, in order to get a more complete approach using variances. Besides, the issue about how to attach these variability concepts to a composition language is still under development.

As well as some tasks can be variable, the number of alternatives may not be static, in such a way that new alternatives might be found at runtime. This issue has not been taken in account, but it could be tackled in later researches allowing adding alternatives automatically as they are detected when new services are discovered by the system.

7 References

Semantics of Interoperable and Outsourced Information Systems

H. Balsters, G.B. Huitema

University of Groningen, Faculty of Management and Organization, P.O. Box 800
9700 AV Groningen, The Netherlands
{h.balsters,g.b.huitema}@rug.nl

Keywords: Information Systems, service delegation, service alignment, semantics, UML/OCL

1 Introduction

Businesses are in nature dynamic and change continuously. Economic prospects can lead to both growth and reduction in size and portfolio. Growth or reduction may consist of outsourcing parts of one’s non-core business processes to specialized parties in the market. In this paper we will concentrate on conceptual modeling of outsourcing information systems, where outsourcing in the context of information systems will be defined as delegating a part of the functionality of the original system to an existing outside party (the supplier). Such functionality typically involves one or more operations (or services), where each operation satisfies certain input- and output requirements. These requirements will be defined in terms of the ruling service level agreements (SLAs). We will provide a formal means to check that the outsourcing relationship between outsourcing party and supplier, determined by a SLA, satisfies specific correctness criteria. These correctness criteria are defined in terms of consistency and completeness between the outsourced operation and the associated operation offered by the supplier. Our correctness criterion will concern mappings between an existing outsourcer schema and an existing supplier schema, and will address both semantical and ontological aspects pertaining to outsourcing. These mappings provide a semantical basis for interoperability between outsourcer and supplier. Our paper focuses on modeling issues; for architectural aspects pertaining to outsourcing of information systems, we refer the reader to [8]. Our emphasis is on semantics of service outsourcing, and not on underlying outsourcing technology. We do, however,
implicitly assume the existence of some typical service-oriented architecture (SOA) as a context in which our work could be placed.

We will focus on the UML/OCL data model to tackle the problem of offering a semantics of outsourcing in the context of interoperable information systems. UML is the de facto standard language for analysis and design in object-oriented frameworks, and is being employed more and more for analysis and design of information systems, in particular information systems based on databases and their applications. The outsourcing relationship will be described in terms of a SLA, containing detailed (and often complex) information on both outsourcer and supplier. Specifying a typical SLA places high demands on the expressiveness and precision of the modeling language employed. We will define so-called exact views \cite{1,3,4} on top of existing information systems in order to eventually capture the formal requirements of the outsourcing relation. A SLA will be given precise specifications in terms of pre- and post-condition statements on operations in OCL, and a correctness criterion will be defined in terms of consistency with respect to these pre- and post conditions.

Our final aim is directed at constructing a mapping from the source model to the target model (i.e. the model of the supplier) preserving the SLA. Such a mapping (called an \(\omega\)-mapping) will be shown to abide to a so-called abstract outsourcing implementation schema (called an \(\omega\)-schema), ensuring correct outsourcing of a source operation. Existence of an \(\omega\)-schema will realize the sought-after (semantical) interoperability between outsourcer and supplier.

This paper is organized as follows. Section 1 offers a description of the correctness problem pertaining to outsourcing. In Section 2 we offer an introduction to views in UML/OCL. Section 3 employs so-called exact views to model outsourcing based on \(\omega\)-schemas and \(\omega\)-maps. This section also contains an illustrative example of an outsourcing relation.

## 2 The problem: Managing correctness of interoperability and outsourcing Preparation

In outsourcing, one typically has the situation that a source company wishes to hand over parts of its functionality to an outside party. This outside party is called the supplier to which the functionality (or service) is outsourced. In terms of models, this situation translates to a (source) model having one or more operations that will be outsourced to an outside (target) model. To be able to perform this outsourcing activity, one is to locate within the source model which operation \(O\) is to be outsourced, along with all relevant attributes, relations, constraints and auxiliary operations that are used in the definition of the operation \(O\). For example, consider the situation of a company wishing to outsource payment and calculation services pertaining to employee salaries. In order to outsource this operation, here called \(calcNetSal\), one might need information regarding present status, date, employee function, employee age, employee address, number of hours that the employee works, initial date of employment, employee department, and possibly more. Once
it has been decided that an operation like `calcNetSal` is to be outsourced, one tries to locate an outside party supplying the functionality of this operation. Once such an outside (target) party is found, the source party and the target party enter negotiations regarding the quality of the outsourcing service that the target has to provide. Once an agreement has been reached, a so-called Service Level Agreement (SLA) is drawn up to which both parties are bound. A SLA is crucial in outsourcing, since it is the sole basis on which source and target parties provide input (responsibility of the source company) and output material (responsibility of the target company). In terms of models, the source model offers the outsourced operation, say $O$, as well as all relevant attributes, relations, constraints and auxiliary operations that are used in the definition of that particular operation $O$. Furthermore, the source model offers its initial conditions that have to be met by a target model. These conditions are the basis for a SLA pertaining to source and target model, and could typically be given in terms of pre- and post conditions. In the case of our example, we could stipulate the following conditions (written in UML/OCL)

```plaintext
context Employee::calcNetSal(presentDate:Date): Integer
pre: status=payrolled
post: age>40 implies result>3000 and result < (self.manager).presentDate.calcNetSal and self.department.depNm=`toy` implies result<4000
```

This specification states that within the class Employee, an operation called `calcNetSal` satisfies the post condition that all employees over the age of 40 earn at least 3000, that no employee earns more than his manager, and that all employees belonging to the department named “toy” earn at most 4000. Should one wish to outsource this operation, then the supplier is bound to this specification. This entails that the supplier is to offer an implementation `calcNetSal'` of `calcNetSal`, such that `calcNetSal'` has a pre- and post condition that are consistent with respect to the pre- and post conditions of `calcNetSal`. For pre conditions this means that `calcNetSal'` should not accept arguments that are not accepted by `calcNetSal`, and for post conditions it holds that `calcNetSal'` should never produces results contradicting a post condition of `calcNetSal`.

In our approach, the SLA between an outsourcing party and a supplier provides the input for a contract binding both parties. The SLA is then used to produce a formal specification, in terms of pre- and post conditions, in which it is precisely (unambiguously) and completely stated what the supplier is expected to deliver. Such a formal counterpart of the SLA is coined a $\sigma$-constraint. The problem that we are now faced with, in terms of models, is to offer a mapping from the model of the outsourcing party (the source model) to the model of the supplying party (the target model) that preserves the $\sigma$-constraint. This mapping serves as a (correctness) specification of the outsourcing relation between source and target. Generally speaking, constructing such a mapping is no trivial matter, and it is the topic of this paper to offer a methodology and a definition of a correct end result for an outsourcing mapping (or $\omega$-map for short). The $\omega$-maps that we are looking for show many resemblances with certain mappings encountered in the field of
data integration. In data integration, we have to map a collection of (local) source models to a (global) target model integrating the source models. Typically, these local models are models of legacy systems, which have to be mapped to a newly defined global system; this problem has been coined as the problem of Global as View ([4,6,7,9,14]). Constructing such mappings from local models to a global model give rise to problems pertaining to so-called data extraction, typically dealing with matters such as naming conflicts (e.g. homonyms and synonyms), conflicts due to different underlying data types of attributes and/or scaling, and missing attributes all deal with differences in structure and semantics of the different local systems. Should we also wish to maintain constraint properties in the transition from local to global, then we move into the realm of so-called data reconciliation ([4,6,10,13]). Data reconciliation is often hard to realize, because of the severe restrictions placed on the mapping from local models to the global model. In [4], it has been shown that only when such a mapping satisfies certain isomorphism properties, correct resolution is ensured of the data reconciliation problem.

In outsourcing we are faced with a more-or-less dual situation: here we are moving from a global system (the system of which a part will be outsourced) to a (possibly collection of) local system(s). Typically, the global system is a legacy system that has to be mapped to an existing local system that will supply the desired outsourcing service. Mapping an existing (global) model to an existing (local) model is known as the problem of data exchange (cf. [10,12]). In general, there are no algorithms for constructing mappings solving the data exchange problem. What we can do, however, as will be done in this paper, is to provide criteria by which it can be judged, in retrospect and after having applied data extraction and reconciliation techniques to semantically align the data schemas on the outsourcer and supplier side, whether the construction of an outsourcing mapping from an existing model to another existing model has been performed correctly. In our case, we offer a criterion, formulated in terms of a so-called Z-schema, judging that a $Z$-map ensures correctness of the outsourcing relation between source and target models. Existence of an $o$-schema will realize (semantical) interoperability between outsourcer and supplier.

We emphasize that in this paper we do not offer a recipe for constructing an $o$-map. What we do offer, however, is a criterion by which we can judge whether our outsourcing has indeed been performed correctly; i.e., by offering a construction of an $o$-map, we offer a guarantee that the outsourcing of the operation will yield results on the supplier side that are completely consistent with the original desired results from the side of the outsourcer. We could also phrase this by saying that the intention of our approach is to guarantee outsourcing transparency. Moreover, we show that a strategy for constructing an $o$-map could consist of devising a specific $\psi$-map that eventually will yield the desired $o$-map.

The next section is devoted to views in UML/OCL; views will be used in subsequent sections of this paper for constructing exact views and $o$-maps in the context of outsourcing.
3 Views in UML/OCL

Consider the case that in the context of some company, we have a class called Emp1 with attributes nm1 and sal1, indicating the name and salary (in euros) of an employee object belonging to class Emp1

\[
\begin{array}{|c|}
\hline
\text{Emp1} \\
-\text{nm1} : \text{String} \\
-\text{sal1} : \text{Integer} \\
\hline
\end{array}
\]

(1)

Now consider the case where we want to add a class, say Emp2, which is defined as a class whose objects are completely derivable from objects coming from class Emp1, but with the salaries expressed in cents. Assume that the attributes of Emp2 are nm2 and sal2 respectively (indicating name and salary attributes for Emp2 objects), and assume that for each object e1:Emp1 we can obtain an object e2:Emp2 by stipulating that e2.nm2=e1.nm1 and e2.sal2=(100 * e1.sal1). By definition the total set of instances of Emp2 is the set obtained from the total set of instances from Emp1 by applying the calculation rules as described above. Hence, class Emp2 is a view of class Emp1, in accordance with the concept of a view as known from the relational database literature. In UML terminology [15], we can say that Emp2 is a derived class, since it is completely derivable from other already existing class elements in the model description containing model type Emp1. Class Emp2 can be described as a derived class in UML/OCL [2,15] in such a way that it satisfies the requirements of a (relational) view. The set of instances of class Emp2 is the result of a calculation applied to the set of instances of class Emp1. The basic idea is that we introduce a class called DB that has an association to class Emp1, and that we define within the context of the database DB an attribute called Emp2. A database object will reflect the actual state of the database, and the system class DB will only consist out of one object in any of its states. Hence the variable self in the context of the class DB will always denote the actual state of the database that we are considering. In the context of this database class we can then define the calculation obtaining the set of instances of Emp2 by taking the set of instances of Emp1 as input.

\[
\text{context DB} \\
def: \text{Emp2}: \text{Set(Tupletype{nm2:String, sal2: Integer})} = \\
(self.emp1-> \text{collect}(e:Emp1 | \\
\text{Tuple\{nm2=e.nm1, sal2=(100*e.sal1)\}}))-> \text{asSet}
\]

(2)
In this way, we specify Emp2 as the result of a calculation performed on base class Emp1. Graphically, Emp2 could be represented as follows where the slash-prefix of Emp2 indicates that Emp2 is a derived attribute

### Exact views

In the case that a derived class in UML is the result of a so-called \( \psi \)-map [3,4], then we speak of an exact view. Exact views belong to the domain of data extraction, and are constructed from a certain collection of injective conversion functions (cf. [3,4]). Exact views have the property that they are correctly updatable, in the sense that any update on an exact view corresponds to exactly one (combination of) correct update(s) on the base class(es) it stems from (cf. [1]).

### 4 Outsourcing by \( \omega \)-schemas and \( \omega \)-maps

Consider the case of our example, described in Section 1. In terms of UML, we can discern within our source model (called SM, for short), the following model elements Manager, Employee and Department. The class Manager is a subclass of Employee, a manager supervises \( n \) (\( n \geq 0 \)) employees, employees belong to exactly one department, and departments have \( m \) (\( m \geq 0 \)) employees. In our example, we (i.e. the company) wish to outsource payment and calculation services pertaining to employee salaries. To be more precise, the company wishes to outsource calculation of the net salary of a given set of employees on the basis of status (pay rolled, or not), present date, employee function, employee age, number of hours that the employee works, initial date of employment and employee department. In terms of SM, not all elements of the class Employee are relevant for outsourcing of these operations. To this end, we will construct a view on the class Employee containing, in general, only those attributes, relations, operations, and constraints that are relevant for our particular outsourcing application. This view /SalView (called a source view) is depicted below

<table>
<thead>
<tr>
<th>/SalView</th>
</tr>
</thead>
<tbody>
<tr>
<td>status: Set(payrolled, notPayrolled)</td>
</tr>
<tr>
<td>function: String</td>
</tr>
<tr>
<td>age: Integer</td>
</tr>
<tr>
<td>hours: Integer</td>
</tr>
<tr>
<td>employedSince: Date</td>
</tr>
<tr>
<td>depNm: String</td>
</tr>
<tr>
<td>man-id: Integer</td>
</tr>
</tbody>
</table>

\[ \text{calcNetSal(presentDate:Date): Integer} \]