

INTEROP deliverable DTG 6.2 : Method repository

Citation for published version (APA):

Jeusfeld, M. A., Ralyté, J., Arni-Bloch, N., Lillehagen, F., Kühn, H., Goossenaerts, J. B. M., Elvesaeter, B., Backlund, P., & Norberg, R. (2006). *INTEROP deliverable DTG 6.2 : Method repository*. Interop Network of Excellence.

Document status and date:

Published: 01/01/2006

Document Version:

Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy

If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.



Network of Excellence - Contract no.: IST-508 011
www.interop-noe.org

Deliverable DTG6.2

INTEROP Method Repository

| | |
|-----------------------------|---|
| Classification: | Public |
| Project Responsible: | UNIGE #52 |
| Authors: | Manfred Jeusfeld, UNTILB, #44 |
| Contributors: | Jolita Ralyté, Nicolas Arni-Bloch, UNIGE #52 Frank Lillehagen, Troux, #55 Harald Kühn, BOC, #28 Jan Goossenaerts, TU Eindhoven, #14 Brian Elvesæter, SINTEF, #16 Per Backlund, Roland Norberg, HS, #48 |
| Task: | DTG6.2 |
| Date: | 2006-11-30 |
| Status: | Final |

Table of contents

| | |
|--|----|
| Deliverable DTG6.2 | 1 |
| Executive Summary | 4 |
| PART I | 5 |
| I.1 Introduction..... | 5 |
| I.2 Methods to produce the deliverable | 5 |
| I.3 Main Results | 5 |
| I.4 Conclusion | 6 |
| PART II | 8 |
| II.1 Introduction | 8 |
| II.2 Presentation of cases..... | 8 |
| II.2.1 Insurance Case..... | 8 |
| II.2.1.1 Business Model | 8 |
| II.2.1.2 Interoperability Issues in the Strategic Business Domain | 9 |
| II.2.1.3 Interoperability Issues in the Operational Business Domain | 10 |
| II.2.1.4 Interoperability Issues in the ICT Domain | 11 |
| II.2.1.5 Summary of the Case | 11 |
| II.2.2 Public Utility Sector Case..... | 11 |
| II.2.2.1 Interoperability Issues in the Business Domain | 12 |
| II.2.2.2 Interoperability Issues in the ICT Domain | 12 |
| II.2.2.3 Summary of the Case | 13 |
| II.2.3 Model-driven software development case – ATHENA MDI framework | 14 |
| II.2.3.1 Sub-case 1: Reengineering and development of MDD guidelines | 15 |
| II.2.3.2 Sub-case 2: Reengineering of the COMET methodology..... | 16 |
| II.2.3.3 Sub-case 3: Development of new method chunks for SOA interoperability | 17 |
| II.2.3.4 Sub-case 4: Prototype implementation with the Eclipse Process Framework (EPF) | 18 |
| II.2.4 Lessons learned..... | 19 |
| II.3 Notion of a reusable method chunk | 19 |
| II.3.1 Definition, Metamodel..... | 19 |
| II.3.2 Guidelines for method chunks definition | 21 |
| II.3.3 Metis-based representation of method chunks | 22 |
| II.3.4 Examples of Method Chunks for Interoperability | 23 |

| | | |
|----------|--|----|
| II.3.1.1 | Method Chunk: Product Process Dependency | 23 |
| II.3.1.2 | ICT Method Chunk: B2B Architecture | 24 |
| II.3.1.3 | Metis representation of a chunk example..... | 25 |
| II.4 | A metamodel for interoperability classification framework..... | 26 |
| II.4.1 | Ontological dimensions for classifying an interoperability problem | 26 |
| II.4.2 | Classifying method chunks..... | 27 |
| II.4.3 | Classifying application cases..... | 28 |
| II.5 | Repository of method chunks | 29 |
| II.5.1 | Case Metamodel | 30 |
| II.5.1.1 | Sample Case template | 31 |
| II.5.2 | Enterprise Architecture and Project Charters of the MCR..... | 32 |
| II.5.2.1 | Society Scale Architecture Descriptions and Project Charter | 33 |
| II.5.2.2 | Organization-scale Architecture Descriptions and Project Charter | 36 |
| II.5.2.3 | Person-scale Architecture Descriptions and Learning Targets | 38 |
| II.5.3 | Technical Architecture of the MCR | 39 |
| II.5.3.1 | Use case descriptions | 39 |
| II.5.3.2 | Service architecture | 41 |
| II.5.3.3 | Provided services..... | 41 |
| II.5.4 | Implementation description | 42 |
| II.6 | Conclusion | 43 |
| II.7 | References | 43 |
| PART III | : Appendices..... | 46 |

Executive Summary

This deliverable presents the INTEROP method chunks repository (MCR), its architecture and provided services. It includes the definition of a reusable method chunk, its structure, illustrated with examples of method chunks stored in the repository and guidelines for method chunks definition and characterisation covering tasks TG6.2 and TG6.3 of the work plan of the task group.

The main result is the definition of the structure of the method chunk repository emphasizing the link to interoperability. Interoperability is a first-class concept in the structure of the method chunk repository. It not only characterizes method chunks, i.e. procedures to solve interoperability problems, but also interoperability cases, i.e. the presentation of actual problems involving interoperability issues.

TG 6 has produced three MCR prototypes. Two experiments were undertaken using the Metis system and one using ConceptBase. The task group attended a two-day intense workshop on Metis. As a result, two experiments with Metis as platform for the method chunk repository are under way and reported in this deliverable. One is realizing the structure of the MCR as specified in this report. The other is an alternative approach that serves as a benchmark and is reported in the appendix. The ConceptBase prototype utilizes the metamodel presented in this deliverable.

We have analysed three cases involving various aspects of interoperability. One case is about establishing a broker platform for insurance agents, the second about linking the information systems in the public utility sector, and the third case is establishing the relation of the ATHENA Model-Driven Interoperability Framework to the goals of the MCR.

The results of the TG6 have been published at the ISD conference 2006 and the ER conference 2006. Copies of the papers are included in the appendix.

The report of the example session with the method chunk repository has been shifted towards deliverable TG6.3 (Tutorial of the MCR). This is the more logical place. We want to emphasize that TG6 was not only busy in drafting concepts, exploring the state of the art, and analyzing cases. We are actually experimenting with a prototype and consider this a valuable contribution to the network. As soon as the prototype is stable, knowledge about interoperability solutions can be coded in this repository and can guide designers of interoperable systems by experience knowledge.

PART I

I.1 Introduction

The task group 6 has an ambitious goal to design a method chunk repository, i.e. a system in which interoperability cases can be matched against methods to solve the problems in the cases. The deliverable DTG6.1 has reported the state of the art in method chunk definition and characterization of interoperability problems. The deliverable DTG6.2 analyses realistic cases to extract from them the definite structure of the method chunk repository, called method chunk metamodel. Additionally, we derive from the cases a multi-dimensional characterization of interoperability problems that is used to index both method chunks (i.e. solutions to interoperability problems) and interoperability cases (i.e. aggregations of several interoperability problems as occurring in practice).

I.2 Methods to produce the deliverable

The task group followed a strategy of intense workshop meetings during the INTEROP meetings and additional meetings devoted to particular topics. Notably, the prototypical implementation of the MCR was prepared by a two-day workshop in Skövde in March 2006.

Significant work has been shaped into joint paper writing where the authors had about 5 Skype conferences per paper. The Skype conferences have proven to be a very productive and cost-effective tool to discuss the distribution of work among collaborators and to resolve open questions.

The overall strategy of the task group is to assess the feasibility of a method chunk repository by implementing a partial prototype. The implementation does not require writing program code but to configure an existing tool capable of metamodeling. The group has looked at various alternatives and selected the Metis tool [37]. By configuring the tool to be a method chunk repository, the task group also learned about how the abstract concepts for interoperability are related to existing modelling constructs of a rich modelling environment such as Metis.

I.3 Main Results

We achieved the following results in the last period reported by this deliverable.

1. We analysed three interoperability cases. Two of them are from industrial practice and one is from the sister project ATHENA. The two industrial cases are also reported in papers published by members of the task group (see appendix 3 and 4). The main finding of this work was that interoperability cases are composed of a multitude of interoperability problems that are rooted in the business domain (either design or execution) and in the ICT domain (also design or execution). Hence, the MCR must support approaches to tackle problems from both domains in an integrated fashion. The cases are reported in section II.2.

2. We derived from the cases the final structure for the method chunk repository, encoded as metamodel (see section II.3). The metamodel extends prior approaches by linking method chunks explicitly to interoperability problems. A simple template for describing method chunks is derived from the metamodel. The concept of interoperability has been subject of intense discussion on the task group and we finalised a multi-dimensional structure for interoperability classification (section II.4). A potential extension is discussed in section II.5.
3. We investigated the architecture of the MCR using the COMET [30] method as guideline (section II.6). The method yields a list of usage scenarios of the MCR.

The partial **Metis-based prototype** has been presented by screen dumps to the INTEROP community during the Bergen workshop in May 2006. Two other prototypical implementations were undertaken, one also based on Metis but with a different metamodel. The third, using the same metamodel, is not based on Metis but on ConceptBase.

I.4 Conclusion

The TG6 has achieved its goal of designing the method chunk repository and has exceeded its original work plan by realizing two (still limited) prototypes based on the Metis system and one using ConceptBase. The idea of a method chunk repository for interoperability has been accepted by the scientific community as well as by industrial partners. In particular, Troux Technologies contributed their Metis system [37] to the task group because they identified the gap of method support in their tool. Likewise, the partner BOC has closely been working with the task group because they want to learn how to enrich their Adonis system by facilities of a method chunk repository.

In the wider INTEROP community, the idea of explicitly storing method chunks in a system has also been adopted by other task groups, for example in the task group 3 and in the Domain OM. The multi-dimensional characterization of interoperability problems is shared with Domain DI. As a consequence, work done in these groups naturally fits into our design of the method chunk repository. The design is finalized and validated by a partial prototype. It is thus ready to be uptakes by either commercial vendors or research projects following INTEROP.

The task group shall continue its work by creating a tutorial and publishing results in a high-quality journal.

This deliverable is organised as follows. Part II presents:

- three cases which show how method chunks can be produced in a practical context,
- the method chunk metamodel and an illustrate by means of presenting two method chunks,
- the Metis representation of a method chunk,
- the classification framework for method chunks and
- the enterprise and technical architectures of the MCR

Part III includes the following appendices:

- A1. The multi-scale COMET
- A2. Paper presented at the ISD conference 2006
- A3. Paper presented at the ER conference 2006
- A4. Benchmark Metis MCR prototype
- A5. ConcpetBase MCR prototype

PART II

This part contains the various documents produced by the various tasks in order to prepare the deliverable.

II.1 Introduction

The main results of this deliverable are organised in four chapters. In chapter II.2 we analyse three interoperability cases and identify interoperability problems. Two cases concern industrial practice while the third one is related to the ATHENA project. In chapter II.3 we define the notion of the reusable method chunk, provide its metamodel and illustrate with method chunk examples. Chapter II.4 presents the interoperability classification framework and its metamodel that we use for interoperability problems classification and method chunks characterisation. Finally, the architecture of the method chunk repository (MCR) is presented in chapter II.5.

II.2 Presentation of cases

In this chapter we present three interoperability cases and identify interoperability problems in each of them. The first case deals with the development of a B2B platform in the insurance domain supporting collaboration of different partners such as insurance companies and independent agents. The second case analysis the public utility sector in the Netherlands, in particular the water sector, and identifies the interoperability problems in this domain. These two industrial cases are reported in co-authored papers published in the international conferences ISD'06 and ER'06 (see appendix 3 and 4). Finally, the third case analysis how the approach proposed by the TG6 could be applied to the ATHENA model-driven interoperability (MDI) framework developed in the ATHENA project.

II.2.1 Insurance Case

In this section we analyse an industrial case of interoperability in the insurance domain and identify interoperability problems related to this case. To structure these problems we use the strategic business domain, the operational business domain, and the ICT domain including development and execution aspects, as proposed in [33].

II.2.1.1 Business Model

Insurance companies develop business models based on Internet technology either to reduce administration costs or to establish new sales channels. They have to establish a well-defined strategic position in the network of their competitors - especially when they join together to establish a common Internet platform for their sales partners, e.g. agents and brokers, to share platform development and operation costs.

The following industry case describes a *B2B sales platform for insurance partners based on Internet technology* ("insurance portal"). The main objective of the insurance portal is to

support independent insurance agents with a single point of access to products and services of different insurance companies. An agent is working for several competing insurance companies on a commission basis. Some advantages for the agents are a single point of access to reduce cycle times for business processes such as offer management, contract management, and portfolio management, less administration costs, and improved service quality because of a broad product and information portfolio. Some advantages for the insurance companies are reduced maintenance and operation costs for their partner systems due to cost sharing and an enlarged sales force because of potentially new agents.

Figure 1 describes the business model of this industry case, i.e. how the different business participants interact with each other to create business value.

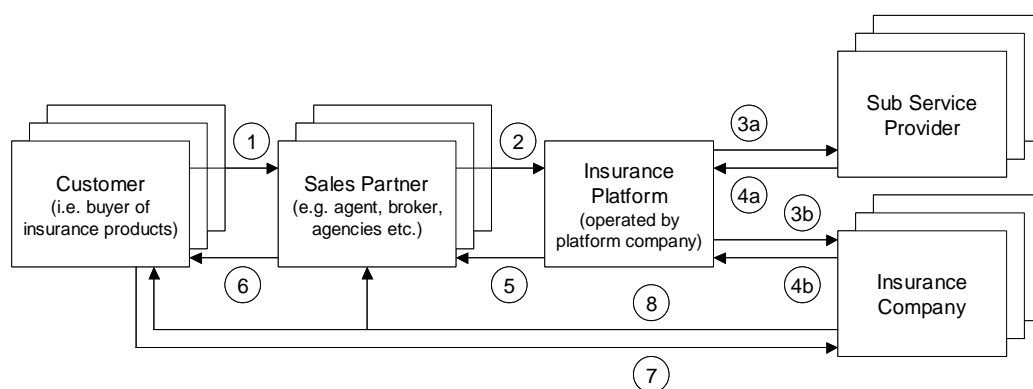


Figure 1. Business Model from Insurance Domain based on Common Platform

Customers interact with their *sales responsables* e.g. agents, brokers, agencies etc. (step 1). A sales responsible uses the insurance portal to execute his business processes such as offer management, order management, policy management etc. For example, a broker may request certain product offers (step 2) which are calculated and returned to him (step 5), and then sent to a customer (step 6). The insurance portal, or more precisely the company operating the platform, interacts with different *sub providers* such as application hosting companies, security companies, customer information suppliers etc. to fulfil its tasks (steps 3a and 4a). Additionally, the company operating the platform interacts with the *insurance companies* to exchange product data, customer data etc. (steps 3b and 4b). Finally, the customer signs a contract with the insurance company, which provided the best offer, and pays the insurance fee to the insurance company (step 7). The insurance company delivers the appropriate contracts, pays the commission fees, and fulfils its part of the insurance contract (step 8).

II.2.1.2 Interoperability Issues in the Strategic Business Domain

In the *strategic business domain*, the business strategy of each participating partner has to be defined in the context of the insurance portal and interoperability questions such as the following have to be answered:

- *Which are the processes and services (products) to be realised on the platform?* Processes, services (products) and their interdependencies have to be identified. Intra-organisational business processes (e.g. user management on the platform) and inter-organisational business processes (e.g. application and claims processes) can be distinguished.

- *Which are the appropriate business partners to develop and run the platform?* According to the required processes and services (e.g. insurance core services, consulting services, implementation and provider services) partners are involved with different contractual relationships (e.g. associate, supplier, customer etc.).
- *Does the business plan of the platform correspond with the business plans of each partner?* Each partner has to agree upon the platform strategy. For example, the standardisation of strategies of competitors participating in the platform may imply the request of investigation of antitrust law. Furthermore, advantages realised by one partner may damage business of another partner (e.g. insurance company A delivers a particular insurance policy within one day, insurance company B in seven days).

II.2.1.3 Interoperability Issues in the Operational Business Domain

In the operational *business domain* the various types of processes have to be determined. The business processes have to be modelled in detail with a special focus on the products and interfaces between the business actors involved. The roles of each business actor also have to be modelled. Business processes can be divided into the following types:

- *insurance core service processes*, e.g. application processes and claims management,
- *value adding processes*, e.g. cash management processes and event management,
- *development processes*, e.g. business and software development based on the core elements: products, processes, organisational units and information technology,
- *business operations processes*, e.g. process integration of business partners and
- *additional services*, e.g. legal advisor services, training and learning.

The following list shows some areas of interoperability problems and opportunities in the business domain:

- *Product Management:* In every realisation state a set of products is integrated into the platform, which entails new requirements for the business processes. Implications for the software development and integration efforts of the insurance partners should be evaluated as early as possible.
- *Process integration of business partners:* Each actor participating in the platform realisation can be certified with respect to its business processes. Some criteria are complexity of interfaces (business operations as well as data flow), process benchmarks, availability and integrity.
- *Training and Learning:* Business processes can be documented online for learning the sequence of operations of core processes as well as administrative processes.
- *Pricing Model:* Agents pay for using the insurance portal. If insurance companies want to consolidate their customer database, the platform company can reduce the cost of the business process “Customer Data Modification” to encourage the agents to reach insurance partners objectives.
- *Test Management:* In combination with the product model, a set of test cases can be developed as a specification for testing the platform application and interoperability.

II.2.1.4 Interoperability Issues in the ICT Domain

The *ICT domain* is divided into *development issues* and *execution issues*. The insurance portal consists of a core service application, dynamic HTML-based user interface, complex application modules etc. During platform *development* typical interoperability problems are:

- How can the different viewpoints of requirement definition be integrated e.g. how can the metamodels of the specification models be integrated?
- Which implementation technologies and target platforms will be used and how will they be integrated?
- What are the different modules of the implementation environment and how can they be integrated?
- Which runtime libraries can be used and how can they be bound to the development environment?

The *execution* domain is influenced by short release cycles - especially driven by short term content such as news and events and by a high fluctuation of platform users. Business operation processes such as content management processes, user management, and first and second level support, are documented by exporting all required information in a process-based online operating instruction manual. Some interoperability problems in the execution domain are:

- Data conversions: Customer data, contract data, product data etc.
- Component integration: How can different components of functionality be operated within a single business service (even if they are realised with different technologies)?
- How can long lasting transactions be synchronised and consistently integrated?

II.2.1.5 Summary of the Case

The above case study is based on a real industrial project. It shows that an ICT project integrating several organisations is typically characterised by a multitude of interoperability problems, in our case totalling to about 20. It also shows that a purely ICT-based answer to the interoperability problem is not only insufficient but also misses the fact that first one has to solve the business-related interoperability problems before one can tackle the ICT-related issues. A consistent method that will solve all possible interoperability problems does not exist because the business and ICT domains are too diverse. Instead of a single method, an extensible and domain-specific knowledge base of method chunks shall support the development of interoperable systems.

II.2.2 Public Utility Sector Case

A typical case from the real world contains multiple interoperability issues. We use as an example the experience from the public utility sector in the Netherlands, in particular the water sector consisting of organisations that *supply fresh water*, organisations that *process sewage water*, and local municipalities that *raise taxes* on both, in particular wrt. the sewage water. In the Netherlands, fresh water supply and sewage water processing are done by organisations that did not have a need to exchange data since the cost for fresh water supply is

based on consumption whereas the cost for sewage water is based on the number of persons in a household. A European guideline stimulates countries to base the sewage water invoice on consumption as well. Since there are no metering devices installed for sewage water per household, the only way to do so is to rely on the metering for the fresh water consumption of the household. To complicate the situation further, the local municipalities use to include taxes on the sewage water invoice that are currently based on sewage water price. As the computation of the sewage water price changes, the tax calculation has to change as well. In the following, we analyse the interoperability problems occurring in the case and classify them into our framework.

II.2.2.1 Interoperability Issues in the Business Domain

Interoperability Problem 1: The *business models* of the three organisations are incompatible. The fresh water organisation raises income based on the consumption. The sewage water organisation and the local municipalities use number of persons in a household as basis for their invoice. Moreover, the participating organisations have different concepts for the addressee of the invoice. The fresh water organisation has a concept of a customer linked to a fresh water supply end point. The other two organisations use the concept of a household with a number of citizens associated to it. To integrate the business models of the three organisations, one needs to come up with calculations on a common data basis that fulfils the expectations of the three organisations.

Interoperability Problem 2: The *business processes* of the three organisations are not aligned. In particular, the invoicing processes are taking place at different points of time. Specifically, the time when a fresh water invoice is printed is completely independent from the time when the sewage water invoice is printed. The processes for maintaining the customer and citizen data sets in the participating organisations need to be aligned since it may well be that a person is still in the customer data set of the fresh water organisation while already being removed from the citizen data set.

Interoperability Problem 3: The *cultural background* and habits in the three organisations is different und difficult to harmonise. The non-profit character of the local municipalities may clash with the more commercial attitude found in the fresh water company. The challenge is to make the right people communicate and exchange information about their respective goals and capabilities. A further complication is that the cooperation is forced upon the participating organisations by the European directive.

II.2.2.2 Interoperability Issues in the ICT Domain

Interoperability Problem 4: The three organisations use completely heterogeneous *IT infrastructures*. The data exchange between the local municipality and the sewage water organisation is done by physically sending spreadsheet files on computer-readable media. The fresh water organisation relies on an ERP system to manage all its data and processes. It is unclear whether to use a common platform to which all three organisations supply data, or to send data directly to each other, or for one of the three organisations to play the role of a data integrator.

Interoperability Problem 5: The *data structures* are heterogeneous. That holds for all fields relevant for creating the invoices, e.g. the address field, the date field etc. The heterogeneity is resolved by ad hoc procedures to reformat the exchange files. For those parties that do not yet exchange data, the problem of heterogeneity is not yet analysed.

II.2.2.3 Summary of the Case

Table 1 classifies the five identified interoperability problems into the framework presented in chapter 2.4. The classification of the case problem is a manual process and is the first step of the MC enactment and cases solutions service of the MCR. The classification limits the scope of applicable solutions as well as the type of change to be expected from the solution. We applied the following approach for the classification of the case problems:

1. Determine the IS domain of the case problem: The IS domain is characterising the type of knowledge that is necessary to understand the case problem. For example, IP5 belongs to the IS domain 'Development process'. Here, the Swebok [36] knowledge base can be used to characterise the field.
2. Determine the interoperability domain: This classification characterises the type of interaction that causes the case problem. For example, IP2 is about the alignment of business processes of operational users at different enterprises.
3. Determine the interoperability class: This class is specifying which type of management activity is related to the interoperability problem. It also specifies which expert is to be consulted to solve the problem.
4. Determine the interoperability issue: The set of issues is build upon experience, i.e. whenever a case problem occurs one looks up whether there is a similar issue in the method chunk repository. The issues are the most specific abstractions of past case problems. The interoperability issue is the item that is linked to the potential solutions in the method chunk repository.

This stepwise approach focuses the case classifier (i.e. the user who describes an interoperability case to the MCR) towards the most relevant interoperability issue for the case problem to be classified. By linking the case problem to the respective categories, the case user also pre-selects the group of people to be involved in solving the problem at hand. The closer the case user describes the case problem along the 4 categories, the easier is the classification process. We plan to support the classification by a user interface that provides questions for classifying the case into the first 3 categories and then proposes the most applicable interoperability issues. If no issue is found, an update request for the classification manager of the method chunk repository is formulated.

Table 1. Case classification

| Case problem | IS Domain | Interoperability Domain | Interoperability Class | Interoperability Issue |
|--------------|-----------------------|-----------------------------|-----------------------------|--|
| IP1 | <i>organisational</i> | <i>business/strategic</i> | <i>Business management</i> | <i>incompatible business models</i> |
| IP2 | <i>organisational</i> | <i>business/operational</i> | <i>process management</i> | <i>business process alignment, bp interoperability</i> |
| IP3 | <i>organisational</i> | <i>business/operational</i> | <i>knowledge management</i> | <i>organisational culture</i> |

| | | | | | |
|-----|----------------------------|------------------------|------------------------|---|-----------|
| IP4 | <i>IT application</i> | <i>ICT/execution</i> | <i>data management</i> | <i>heterogeneous infrastructures</i> | <i>IT</i> |
| IP5 | <i>development process</i> | <i>ICT/development</i> | <i>data management</i> | <i>data integration, data format interoperability</i> | |

Out of the five identified case problems, three originate from the organisational domain, i.e. require a solution that is not just a technical one. Only problem IP4 apparently requires to change the IT systems, namely to provide the required data in the right format at the right time. By this example we show how the outcome of the case classification is used to search for applicable method chunks in the repository.

We note that a case like the one discussed above touches multiple interoperability issues, which need to be tackled in an orchestrated effort. An open problem is still whether the solution to a complete case should be regarded as a whole, because the solutions to the interoperability problems highly depend on each other, or whether the individual solutions to the individual interoperability problems should be regarded as stand-alone.

II.2.3 Model-driven software development case – ATHENA MDI framework

This section presents the *ATHENA model-driven interoperability (MDI) framework* which has the following objectives:

1. to provide guidelines and method chunks for how MDD principles and MDA technologies should be applied to develop interoperable service-oriented systems;
2. to provide a set of method chunks covering the full software development lifecycle that lets you assemble and configure your situational method; and
3. to provide a set of method chunks focusing on integration and interoperability issues in service-oriented architectures, supporting the use of technologies such as Web services [1] and JACK agents [16].

The ATHENA MDI framework can be seen as an example of applying the whole TG6 method engineering approach (see Figure 2) to the domain of model-driven software development (MDD).

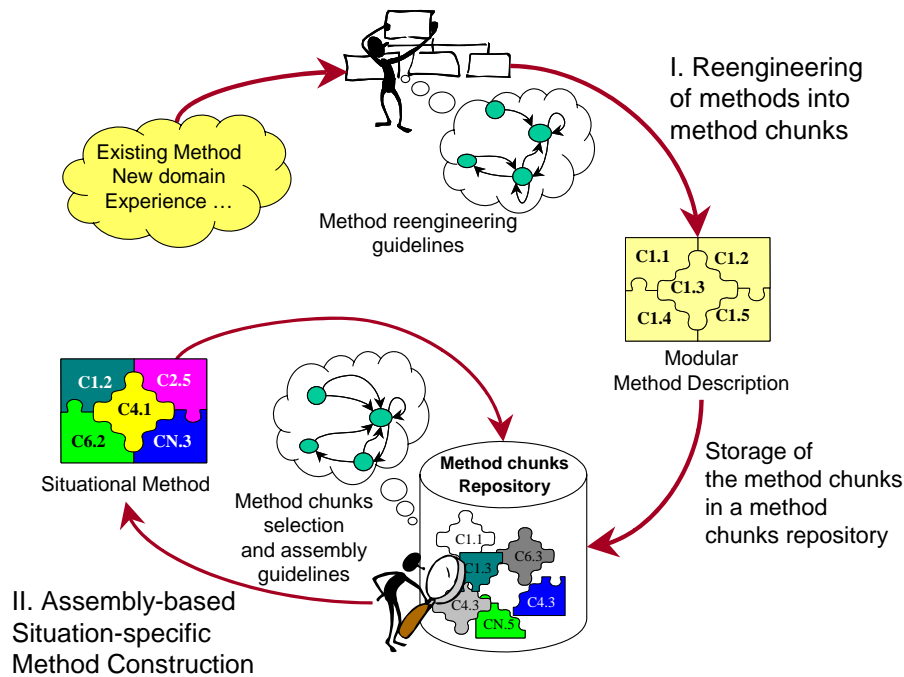


Figure 2. Method engineering process

The development of the ATHENA MDI framework can be divided into four sub-cases covering the two phases of the method engineering process.

| Phase | Sub-case |
|---|---|
| 1. Reengineering of methods into method chunks | <ul style="list-style-type: none"> • Reengineering and development of MDD guidelines. • Reengineering of the COMET methodology • Development of new method chunks for SOA interoperability |
| 2. Assembly-based situation-specific method construction. | <ul style="list-style-type: none"> • Prototype implementation with the Eclipse Process Framework (EPF) |

The following sections describe the four separate cases in more details. We also identify and describe interoperability problems related to each of these cases.

II.2.3.1 Sub-case 1: Reengineering and development of MDD guidelines

The current state of the art in model-driven software development (MDD) is much influenced by the ongoing standardisation activities around the OMG Model Driven Architecture® (MDA®¹) [2, 5]. MDA is a technology framework which defines an approach in which visual modelling languages and visual models can be used to integrate the huge diversity of technologies used in the development of software systems.

The MDI framework is currently being developed as a website [29] which will provide guidelines on the following topics:

¹ Model Driven Architecture® and MDA® are registered trademarks of the Object Management Group

- **Metamodelling:** Guidelines for developing/using metamodels in MOF-based technologies.
- **Language engineering:** Guidelines for developing/using domain-specific languages and UML profiles.
- **Model transformations:** Guidelines for developing/using model mappings and transformations in QVT-based technologies.
- **Method engineering:** Guidelines for developing/using software development methods in SPEM-based technologies.

The MDD approach promises to deliver portable, interoperable and reusable software solutions. We see evidence that interoperability can be supported by model transformation and metamodel alignment using MDA-compliant technologies. However, we also find evidence to suggest that MDA is still immature and needs to be improved and extended in several areas particular with respect to interoperability. The interoperability issues are two-fold, one is the integration of the MDA-compliant technologies themselves, and the other is how to successfully use and extend MDA technologies to develop interoperable systems. The ATHENA MDI framework focuses on the latter. Examples of interoperability-related questions are:

- How to develop metamodels and languages that addresses SOA interoperability issues in software development?
- How to develop (horizontal) model transformations that integrates models defined according to different metamodels?
- How to develop (vertical) model transformations for technology platforms that preserve the interoperability characteristics described at the technology-independent level?

II.2.3.2 Sub-case 2: Reengineering of the COMET methodology

COMET is a software development methodology that is based on the results of the COMBINE (Component-Based Interoperable Enterprise System Development) research project (IST-1999-20839) on component-based software engineering supported by the European Commission. COMET gives us a baseline to support the objective of providing a set of method chunks covering a full software development lifecycle, including business modelling, requirements modelling and technical modelling.

Whereas there exist a number of different development methodologies, e.g. the Unified Process [20], that are more popular and used by the industry on a larger scale, the choice of the COMET methodology provided us with some advantages. The methodology is developed by one of the TG6 partners, it is open and its metamodels are documented. Different variants of the methodology have been used in industry projects, as well as other research projects. During these projects we identified some interoperability issues that suggest that a reengineering into method chunks is required in order to make COMET more flexible and customizable:

- *Issue 1:* Customize and select different parts of COMET. The users only wanted to use specific parts of the methodology, or extend certain parts of the methodology.

- *Issue 2:* Integration with other methods. There are already in place other development methods and techniques that needs to be integrated.
- *Issue 3:* Integration with different tools. Interoperability issues related to the model artefacts prescribed by COMET and the modelling tools already in place in the organisation.
- *Issue 4:* Support different architectural styles. COMET was originally developed to support component-based development. There is now a shift towards service-orientation and it needs to incorporate modelling concepts and modelling techniques for the development of interoperable SOA systems. This issue is related to the sub-case 3 described below.

II.2.3.3 Sub-case 3: Development of new method chunks for SOA interoperability

Service-oriented architectures (SOAs) are an architectural style for distributed systems that has steadily been gaining momentum over the last few years and is now considered as mainstream in enterprise computing. In the transition to SOAs there are numerous interoperability-related questions that must be answered. Examples of such questions are:

- How to recondition the current application architecture and plan for a sustainable service-oriented architecture?
- How to use Web services, Agents and P2P technologies to develop an SOA?
- How to align the SOA with the enterprise architecture?
- How to integrate two or more existing IT services or systems in an SOA?
- How to compose new services from existing services?
- How to design new services? What are the new services needed?
- How to design for interoperability, flexibility and adaptiveness?

In the ATHENA project we have defined an ***ATHENA service-oriented interoperability (SOI) framework*** [22] which provides guidelines for developing and integrating software services in service-oriented architectures. The framework follows the OMG Model Driven Architecture (MDA) approach and defines a Platform Independent Model (PIM) for SOA (PIM4SOA) and Platform Specific Models (PSMs) for describing Web services (XML Schemas and WSDL), Jack BDI agents and BPEL (Business Process Execution Language) processes.

PIM4SOA is a visual PIM which specifies services in a technology independent manner. It represents an integrated view of the SOA in which different components can be deployed on different execution platforms. The PIM4SOA model helps us to align relevant aspects of enterprise and technical IT models, such as process, organisation and products models. This model allows us to raise the abstraction level at which we can talk about and reason on the architecture we design. The framework provides model-to-model transformation services which allow us to transform PIM4SOA models into underlying PSMs such as Web services [5] and JACK agent technology [1].

II.2.3.4 Sub-case 4: Prototype implementation with the Eclipse Process Framework (EPF)

The INTEROP TG6 provides a method engineering (ME) framework that can be used to structure and define reusable method chunks. In the development of the ATHENA MDI framework we have tried to follow the approach of TG6. The intention is to develop three different types of method chunks corresponding to the three objectives of the framework:

1. Example method chunks providing guidelines on how to apply MDD principles and technologies.
2. Example method chunks based on the COMET methodology [23] which will be method chunks for developing baseline software development methods.
3. Example method chunks based on the ATHENA SOI methodology [22] which will be method chunks that addresses how interoperable Web services should be designed, developed and integrated in service-oriented architectures.

The first category of method chunks targets developers of MDD frameworks, i.e. the ATHENA MDI framework itself, with guidelines on how to develop new metamodels, languages, model transformations and method chunks, and populate the method chunk repository with new results.

Method chunks in the two latter categories target assemblers of methods. It allows users to build their own software development method (1), or customise an existing model-driven software method by inserting method chunks addressing specific interoperability needs (2) as illustrated shown in Figure 3.

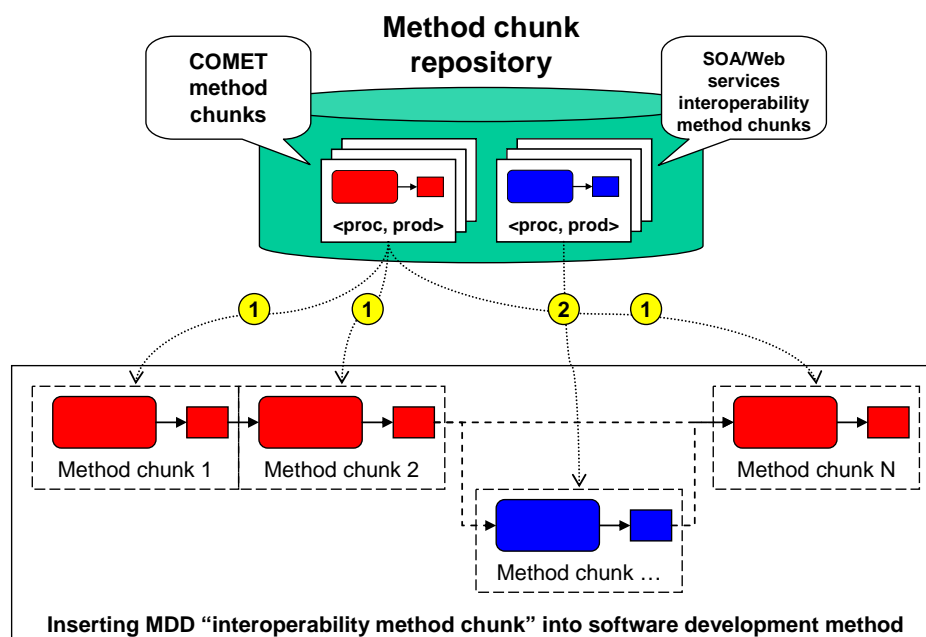


Figure 3. Example usage of the method chunk repository (MCR)

In order to prototype this approach we have experimented with the Eclipse Process Framework (EPF) [8]. EPF is an open-source Eclipse project with the goal of providing an extensible framework and exemplary tools for software process engineering, which includes method and process authoring, library management, configuring and publishing a process. As part of this experiment the COMET requirements modelling method was implemented. A tutorial describing this work is available at <http://modelbased.net/mdi/method/tutorials.html>.

II.2.4 Lessons learned

The industry and the public sector cases show that there is a multitude of interoperability problems situated at the level of business but also ICT. In the industry and public domain cases we identify 25 problems: 16 from the business domain and 9 from the ICT domain. Business and ICT interoperability problems have to be linked.

In this sense these cases are more problem oriented and the diversity of these problems leads to the need of a repository to be able to build situation-based method that response to the specific situation at hand. The MDI case come from the software domain show the need of modular methods to support different type of software projects.

The case inventory and the method reengineering shows the how the concept of method engineering can be applied to the interoperability domain. We also show how results from the ATHENA project can be integrated to an MCR, thus contributing to the integration of research results within the field.

II.3 Notion of a reusable method chunk

The problem of enterprise interoperability is complex and requires support from many methodologies to be resolved, which methodologies are needed depends on the type of system. The future of Systems Engineering will not see just one approach but a multitude of approaches depending on the type of system and the degree of reuse of solutions. Future systems will range from global data collection, analysis and presentation to dynamic systems for mass-customised product design. Therefore, in our work we know that it is impossible to provide one universal approach and set of methods for interoperability problem solving. We therefore propose to define a knowledge base of reusable method chunks each of them addressing one or more specific interoperability problems.

II.3.1 Definition, Metamodel

After the investigation of the role of Method Engineering and the possibilities it offers for constructing new methods in the interoperability domain (see the deliverable DTG6.1) we proposed to use the concept of reusable method chunk to capture method knowledge in systems engineering domain dealing with different interoperability issues and to apply assembly-based method engineering technique allowing users to combine method chunks and therefore to satisfy different project situation.

By a method chunk we mean an autonomous and coherent part of a method supporting the realisation of some specific system development, user application or management activity. In

this work we focus our attention on the method chunks supporting interoperability problems solution.

Figure 4 represents the metamodel of a method chunk. From the Method Engineering perspective each method chunk includes two types of knowledge namely process and product. The process model, also called guideline, supports the engineer in method chunk application while the product model defines concepts, relationships between concepts, and constraints used by the corresponding process. The detailed structure of a guideline can be found in previous publications [32, 34].

The context in which a method chunk is relevant is defined in its interface. It is formalised by a couple $\langle \text{situation}, \text{intention} \rangle$, which characterises the situation in which the method chunk can be applied in terms of required input product(s) and the intention, i.e. the goal, that the chunk helps to achieve.

A set of characteristics, called a method chunk *descriptor*, is associated to each chunk in order to better situate the context in which it can be reused. The *reuse intention* expresses the generic objective that the method chunk helps to satisfy in the corresponding engineering activity. The *reuse situation* captures a set of criteria characterising the context in which the method chunk is suitable. A detailed classification of these criteria, named *Reuse Frame*, can be found in [32]. While the reuse situation and reuse intention are expressed by using keywords defined in the MCR glossary and the reuse frame, the *objective* of the method chunk provides a narrative explanation of its role.

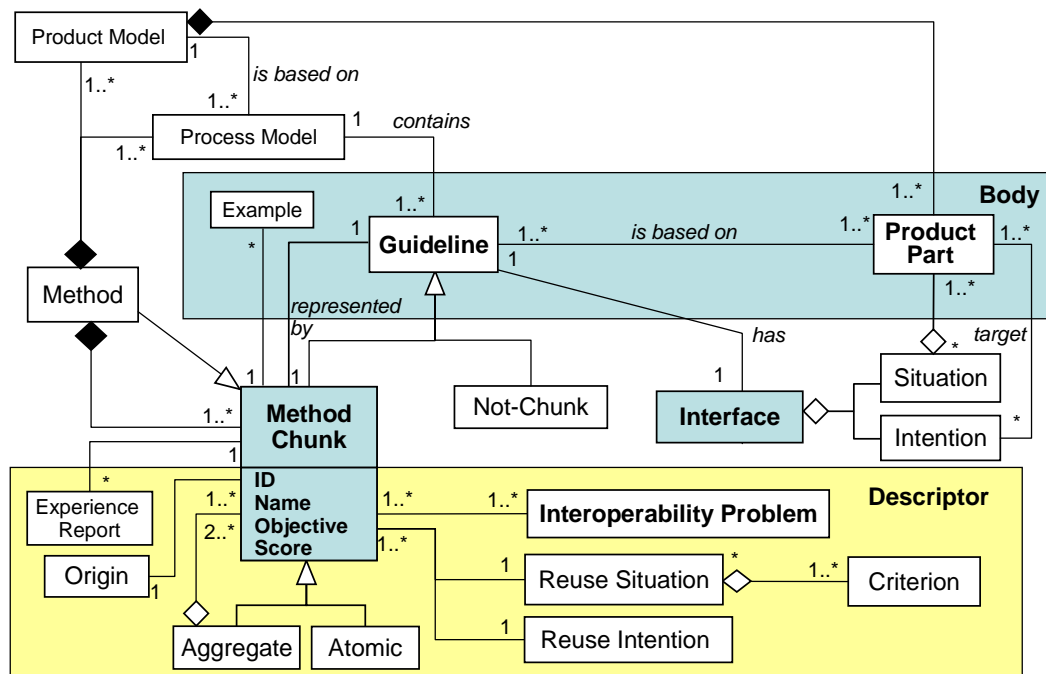


Figure 4. Metamodel of method chunk

Due to the fact that in this work we consider specific method chunks dealing with interoperability problems solution, we explicitly relate each method chunk to the

corresponding *interoperability problem* identified in the interoperability classification framework presented in chapter II.4 of this deliverable.

More details about the notion of method chunk can be found in our previous publications [32, 34] and the co-authored publication [33] included in Annex 4.

We consider that methods, approaches and techniques provided by different INTEROP activities can be defined in terms of method chunks, which could be reused in different interoperability projects and combined in different manners according to the situation at hand.

II.3.2 Guidelines for method chunks definition

A few works related to the SME domain [34, 6, 14, 13] propose to extract method chunks from existing methods by decomposing them. Generally, method chunks illustrated in the literature correspond to different models of traditional methods like the object model, the statechart model or the use case model. According to Ralyté [35] some of these models could be also decomposed into smaller chunks satisfying some more specific goals or re-engineered in another manner and thus could be better situated in the ISD process.

However, the domain of enterprise and IS interoperability is quite new and there is no well known methods supporting it. The INTEROP community is exploring requirements for such methods, approaches and techniques. Our proposal is to define these new methods, approaches and techniques in terms of reusable method chunks. It is clear, that the method decomposition techniques mentioned above are not very adequate. In our case, the process for method chunks construction is mainly Ad-Hoc [35] i.e. based on best practices, experience or new identified requirements.

| | |
|---|--|
| Chunk ID: The unique id of the chunk. | Name: The name of the chunk. |
| Objective: The objective that this chunk aims to reach. | |
| Type: Atomic or Aggregate. | Origin: The existing method or best practice provider. |
| Version: The version number | Authors: The authors of this method chunk. |
| Status: The state of the chunk: In progress, Finished, Deprecated... | Date of creation: The date of the creation. Last modification: The date of the last modification. |
| Interoperability problem: The <i>interoperability problem</i> that this chunk aims to address identified in the interoperability classification framework (see interoperability classification framework). | |
| Reuse situation: Captures a set of criteria characterising the context in which the method chunk is suitable. See the classification framework. | |
| Reuse intention: Expresses the generic objective that the method chunk helps to satisfy in the corresponding engineering activity. | |
| Interface: Situation: The situation in which the method chunk can be applied in terms of required input product(s). | |
| Intention: the goal that the chunk helps to achieve. | |
| Body: | |

| |
|---|
| <p>Sub-chunks: The list of sub-chunks if this chunk is an aggregate.</p> <p>Product Part: The product part of this chunk represented in terms of a metamodel and an informal description.</p> <p>Guideline: The process model of this chunk.</p> |
| <p>References: List of references.</p> |
| <p>Application Example: Application <i>examples</i> provided in order to help the method engineer to apply the method chunk.</p> |

Figure 5. A template for method chunk definition

In order to help INTEROP members to create method chunks we provide a template for method chunks definition presented in Figure 5. This template is based on the chunk metamodel presented in Figure 4. As defined in the method chunk metamodel, this template is based on three main parts of the method chunk: the *descriptor*, the *interface* and the *body*. The *descriptor* defines the id, the name, and the objective of the chunk. It also provides some information about the version, date of creation, and the author of the method chunk. Finally the *descriptor* specifies the interoperability problem that this chunk addresses and the reuse situation and intention. The context in which a method chunk is relevant is defined in its *interface* as a couple $\langle \textit{situation}, \textit{intention} \rangle$ as defined in the metamodel (Figure 4). Finally the body of a method chunk includes the method knowledge (guidelines and product definition) to be applied in the specified context.

II.3.3 Metis-based representation of method chunks

The method chunk template is defined for textual method chunks representation. This kind of representation has the advantage of the simplicity but has a lack of formal structuring that generates some drawbacks for method engineering (as for example during the assembly process). To better support the process of method chunk engineering we provide a Metis representation of the method chunk. The Metis tool is one of the so-called meta-CASE tools. It provides an easy way to define our own metamodel and a modelling representation to manipulate this metamodel. The Metis tool also supplies a collaborative repository that can serve as a first method chunk repository (MCR).

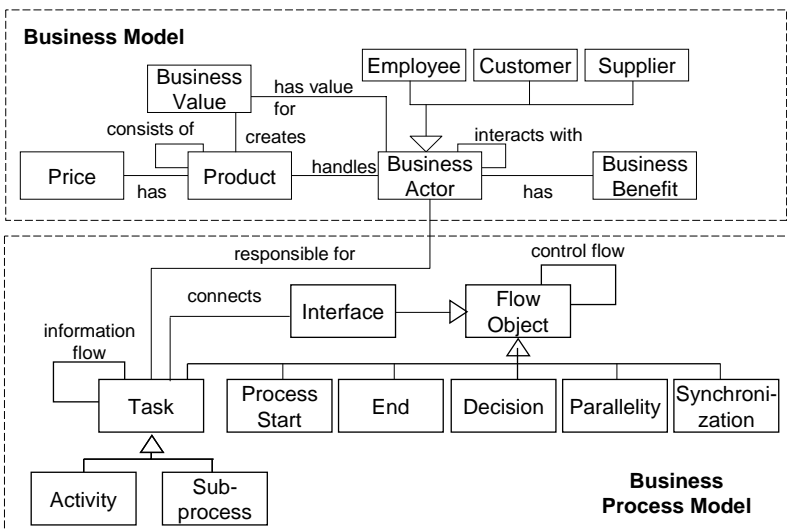
Based on the method chunk metamodel presented in Figure 4, we have implemented the Metis representation of a method chunk. We have first defined the body of the chunk, i.e. the process and product parts, using a process-data diagram. In this diagram, the product part is specified by using a UML class diagram and the process part using a UML activity diagram. The two are linked with a *produce* link between an activity and a class. These two parts compose the core of a method chunk (Figure 8). In addition to the standard definition of the UML Activity diagram, we have added the possibility to link an Action to a method chunk (with a Use Chunk link). This allows us to represent aggregated method chunks and to clearly situate sub-chunks in the process.

II.3.4 Examples of Method Chunks for Interoperability

Based on the practical experiences in the insurance case presented in chapter II.2.1 we have identified several method chunks dealing with interoperability problems in this case. Due to the lack of space, we present only two of them: one from the strategic and operational business domain and one from the ICT domain. To define each method chunk, we use the template introduced in section II.3.3.

II.3.1.1 Method Chunk: Product Process Dependency

Different enterprises form a supply chain and they have to align their products and their business processes. It must be defined which products and product definitions are interrelated with which processes and process interfaces. The method chunk below (Figure 6) proposes a solution for this kind of interoperability problem.

| | |
|--|---|
| Chunk ID: MC01 | Name: Product Process Dependency |
| Objective: Identify dependencies between products and their corresponding business processes as basis for business alignment. | |
| Type: Aggregate | Origin: BOC Information Systems |
| Interoperability problem: Business.Strategic_and_Operational.Business Alignment | |
| Reuse situation: Application domain.Application type.Inter-organisation application Application domain.Impact of legacy system.Functional domain reuse System engineering activity.Business modelling.Business process alignment Innovation level.Business innovation | |
| Reuse intention: To align product definitions and business process definitions. | |
| Interface: Situation: Products and business processes of partner enterprises. Intention: To define integrated product and process modelling language. Body: Product Part: Integrated definition of products and business processes. | |
|  | |
| Guideline: Define the product structure in accordance with the business metamodel. Define the business process structure. Assign the responsible business actors to the activities and sub-processes of the business process. Define the interfaces which are necessary to connect the activities and sub-processes. By assigning | |

the product responsibilities between products and business actors, the dependencies between products and business processes are defined transitively.

Application Example:

An application example of this method chunk is the definition of insurance products and their interdependency to business processes executed in the insurance portal. A life insurance product consists of sub-products such as risk insurance and font investment. A life insurance process consists of sub-processes such as insurance application, risk check, contracting and payment. Employees of insurance companies are responsible for executing the sub-processes. These employees are also handling several insurance products. Via this, the product process dependency is defined.

Figure 6. Method Chunk dealing this Business Product and Process Dependency

II.3.1.2 ICT Method Chunk: B2B Architecture

Different companies want to establish a common Internet-based platform implementing parts of their e-business processes. The existing company strategies, business processes and information systems have to be interoperable with this new platform.

| | |
|--|--|
| Chunk ID: MC02 | Name: B2B Architecture |
| Objective: To provide a general architecture for a collaborative Internet-based partner platform. | |
| Type: Atomic | Origin: BOC Information Systems |
| Interoperability problem: ICT. Development. B2B Architecture Design | |
| Reuse situation: Application domain. Application type. Inter-organisation application Application domain. Impact of legacy system. Functional domain reuse System engineering activity. Design Innovation level. Technology innovation; Business innovation Reuse intention: To establish a common Internet-based platform. | |
| Interface: Situation: The strategies, business processes and information systems of the involved companies. Intention: To define building blocks for a B2B system. | |
| Body: Product Part: General software architecture of a B2B platform. The arrows depict the different places of interoperability. <div data-bbox="245 1402 1311 1944" data-label="Diagram"> <pre> graph TD subgraph SSP [Sub Service Providers] SSR[Analysis and Retrieval Services] SS[Security Services] CIS[Customer Information Services] Dots1[...] end subgraph PU [Platform Users] WB[Web Browser] end subgraph PC [Partner Companies] CC[Company Components (deployed to platform)] CS[Company Services (integrated into platform)] CD[Company Data (used within platform)] Dots2[...] end subgraph PP [Partner Platform] WSS[Web Server/Servlet Server] ASBS[Application Server/Business Services] PID[Platform Database (internal data)] TED[(Temporary) Database of External Data (e.g. products etc.)] end WB <--> WSS WSS <--> ASBS ASBS <--> PID ASBS <--> TED ASBS <--> SSR ASBS <--> SS ASBS <--> CIS ASBS <--> CC ASBS <--> CS ASBS <--> CD ASBS <--> Dots2 </pre> </div> | |
| Guideline: Identify participants involved in operating and using a B2B platform. For each participant assign which of the generic building blocks are provided/used. Build an instance of each generic building block for | |

the specific case. Describe the interrelationships within the B2B platform for each building block instance.

Application Example: An insurance portal. The identification and assignment is as follows:
Platform users (sales agents, brokers etc.): the sales partners access the portal via Internet and web browser technology.

Insurance partner platform: The access of the business functionality and the generation of the user interface are via web server / servlet server. The business functionality runs on an application server. The application server stores platform internal data in the platform database. External (and temporary) data are stored in the database for external data. Via business services of the application server sub service providers and insurance companies interoperate with the insurance partner platform.

Insurance companies: The insurance companies provide components (e.g. product calculators, risk check modules etc.), services (e.g. printing, mailing etc.), data (e.g. customer data, contract data, product data etc.), which have to interoperate with the insurance partner platform.

Sub service providers: The sub service providers provide services such as analysis and retrieval services (e.g. data analysis, management reports, statistical evaluations etc.), security services (e.g. trust centres certificate management etc.), customer information services (e.g. credit agency services, market evaluation etc.), which have to interoperate with the insurance partner platform.

Figure 7. Method chunk for B2B Architecture definition

II.3.1.3 Metis representation of a chunk example

To illustrate the representation of method chunks with Metis we use the first method chunk example (see Figure 6) identified in the insurance case.

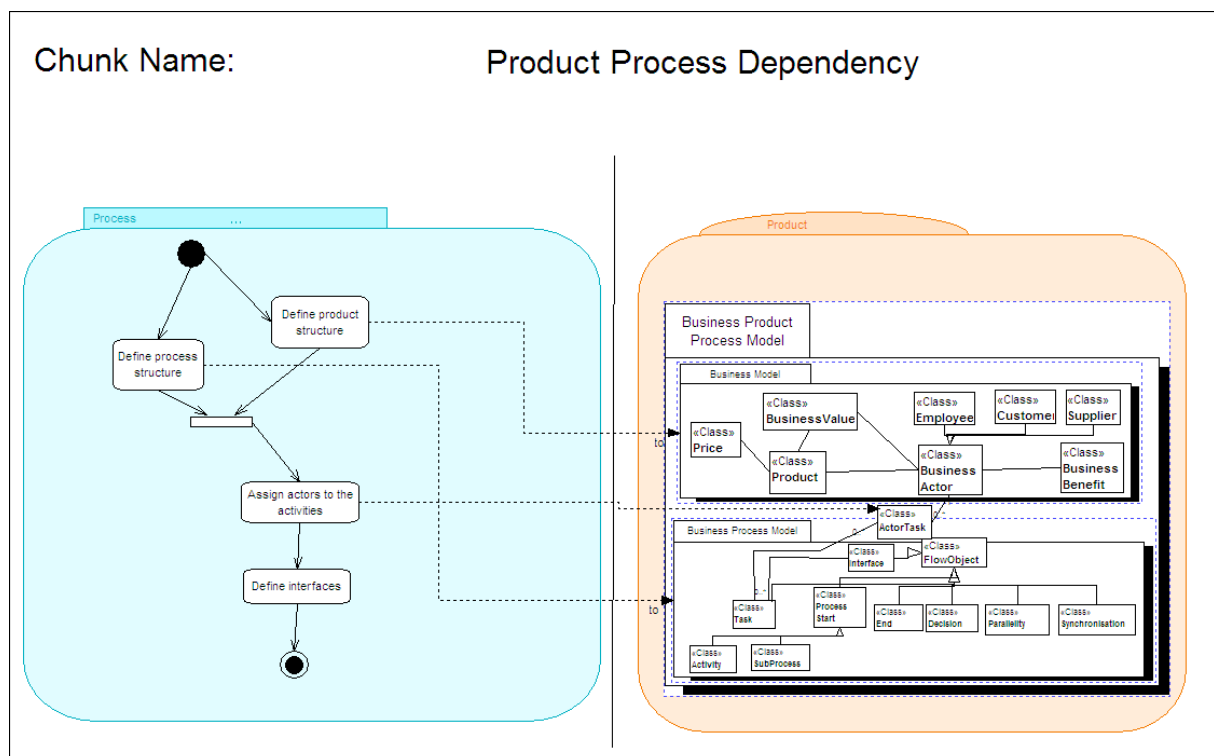


Figure 8. Example of method chunk using Metis representation

In this example, the activity *Define process structure* produces the product named *Business Model* and the activity *Define product structure* produces the *Business Process Model*. These two models must be defined before doing the activity *Assign actors to the activities* which

produces the link between the product and process models represented by the *ActorTask* class. The representation of this method chunk with Metis is illustrated in Figure 8.

II.4 A metamodel for interoperability classification framework

The classification framework has the purpose to associate method chunks as well as cases to re-occurring interoperability problems. By tagging the method chunks with suitable instances of interoperability problems, we index the chunks much like books and articles are indexed in a library: the indexing is supporting the search for method chunks that address a certain interoperability problem. In the same way, actual cases are described in terms of the interoperability problems that are occurring in them. The challenge is to index problems and solutions in such a way that a match between the two is made possible.

II.4.1 Ontological dimensions for classifying an interoperability problem

The concept of a method chunk forms a complementary approach to using patterns as proposed in INTEROP DI (Chen et al., 2006) [7]. Patterns may be stored in a method chunk repository. One advantage of using a ME approach is that patterns will be related to each other as well as to the type of interoperability problems they solve, which will facilitate their use. The problem classifier augments the characterisation of patterns in terms of conceptual, technical and business barriers as proposed by INTEROP DI.

Interoperability problems are occurring in a certain situation within a project concerned with the interaction of multiple organizations and their information systems, hence covering both the business/organizational domain and the ICT domain. The following questions guide the definition of the classification framework:

1. From which knowledge domain can we draw expertise to understand the interoperability problem?
2. During which lifecycle stage does the problem occur?
3. Which types of products are involved in the observed interoperability problem?
4. Which types of processes were active when the problem occurred?
5. Which types of human or automated producers are involved in the problem?

The five questions are translated into classification dimensions as follows.

Knowledge dimension. Iivari et al. [15] propose five ontological domains (Type KnowledgeDomain in Figure 9), which are based on a review of the state of the art in current IS research. These five domains cover the area of Information Systems well. The *organisational domain* refers to the knowledge about social contexts and processes in which the information system is used. Organisational domain knowledge has to do with knowledge management in a general sense even though certain issues may be refined to specific application domains. The *application domain* refers to the knowledge about the application domain for which the information system is intended. Application domain knowledge

includes issues concerning business management and process management, i.e. how typical applications work in a particular domain. The *IT application* domain refers to the knowledge about typical IT applications and their use in a certain application domain. The *technical domain* covers the hardware and software of an information system. In the technical and IT application domains we find issues of data management and software management, hence relating the IS field closely to the field of software engineering. Finally, the *development process* knowledge refers to the methods and tools used in systems development.

Lifecycle dimension. The lifecycle dimension characterises the phase in which some situation is observed or some activity can take place. At the highest level of granularity, we distinguish the phases ‘business-strategic’ (the phase of a project in which strategic business decisions are made), ‘business-operational’ (the phase in which business activities are executed), ‘ict-development’ (the phase in which some ICT solution is developed), and ‘ict-execution’ (the time when some ICT system is performing operations). This level can be further decomposed, for example ‘ict-development.analysis’ is the phase in which the specification of an ICT systems is analysed).

Product dimension. The product dimension specifies types of products that are relevant in some observed situation or that are involved in some activity. Possible values are ‘model-type’ (the involved products have the nature of models), ‘document-type’, ‘notation’, and ‘language’. Like before, specializations are formed like ‘model-type.data-model’ or ‘model-type.source-code.java-program’. For documents, we suggest to form specializations according to the structure of the document, e.g. ‘document-type.contract.sla’ for a service-level agreement.

Process dimension. The process dimension has to be distinguished from the lifecycle phase. It is defined as the processes that are active in some observable situation. At the highest level, we distinguish ‘human-process’, ‘automated-process’, and ‘human-computer-interaction’. At deeper levels, processes like ‘human-process.meeting.group-modeling-session’ are expressed. Another example is ‘automated-process.data-exchange’.

Producer dimension. Producers are human or automated actors that are capable of creating and processing some products. For the purpose of interoperability problem classification, we distinguish ‘role’ (characterising the responsibilities of a human actor, for example ‘role.system-analyst’, ‘team’ (for example ‘team.development-team’), and ‘system’ (e.g. ‘system.tool.diagram-editor’ or ‘system.enterprise-system.crm-system’). Note that producers are observable at any lifecycle stage.

The last four dimensions are adapted from the SMDM standard, which is concerned with describing development processes.

Our metamodel for classifying interoperability problems is represented in Figure 9.

II.4.2 Classifying method chunks

The tagging of method chunks by interoperability problems is the responsibility of the author of the chunk. For standard chunks such as the reverse engineering of a conceptual data model

out of a database schema, the author can create a suitable entry in the list of interoperability problems, e.g. ‘understand legacy databases’. In many cases, a method chunk will be the generalization of successful solution of a case problem. Then, the interoperability problem will have been stored in the MCR as result of classifying a case.

II.4.3 Classifying application cases

A case in the context of the MCR is a situation of a user (or group of users) that includes an interoperability problem that requires to be addressed in a structured way. The classification of the case problem is a manual process and is the first step of the MC enactment and cases solutions service of the MCR. The classification limits the search space of applicable solutions, i.e. method chunks, as well as the type of change to be expected from the solution. We suggest the following approach for the classification of the case problems:

1. Determine the IS domain of the case problem: The IS domain is characterising the type of knowledge that is necessary to understand the case problem. For example, IP4 belongs to the IS domain ‘development-process’. Here, the Swebok [36] knowledge base can be used to characterise the field.
2. Determine the lifecycle stage: Possible values are ‘business-strategic’ (specifying that the interoperability problem encountered is about the business domain and about a strategic decision to be taken by the business partners), ‘business-operational’, ‘ict-development’ and ‘ict-execution’.
3. Determine the involved product types (if applicable).
4. Determine the involved process types (if applicable).
5. Determine the producer type (if applicable): stakeholders, involved organizations, team composition, tools used for production.
6. Determine the interoperability problem: The set of problems is build upon experience, i.e. whenever a case problem occurs one looks up whether a similar problem is already stored in the method chunk repository. The interoperability problems are the most specific abstractions of past case problems. Only the interoperability problems shall be associated to method chunks, i.e. their potential solutions.

This stepwise approach focuses the case user (see Figure 9) towards the most relevant interoperability issue for the case problem to be classified. By linking the case problem to the respective categories, the case user also pre-selects the group of people to be involved in solving the problem at hand. The closer the case user describes the case problem along the 4 categories, the easier is the classification process. We plan to support the classification by a user interface that provides questions for classifying the case into the first 3 categories and then proposes the most applicable interoperability issues. If no issue is found, an update request for the classification manager of the method chunk repository is formulated.

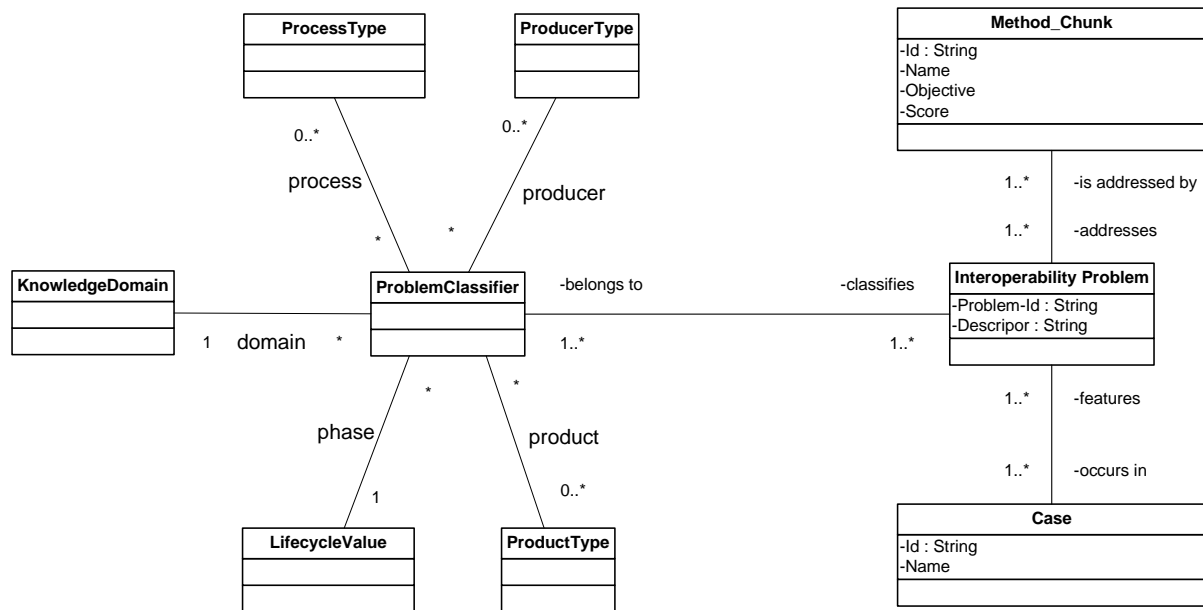


Figure 9. Metamodel for classifying interoperability problems

II.5 Repository of method chunks

This chapter introduces the MCR architecture description that follows the COMET based templates introduced in Annex A1. Remind that MCR services will be delivered to *social networks* of enterprises and private persons via the web and wireless environment. The high autonomy in such networks poses an MCR adaptation challenge. Goossenaerts [12] has introduced some techniques and artefacts for articulating information system (in this case MCR) value propositions to loosely connected actors, for planning system and or function release and for monitoring value-impacts. Those proposals draw on insights from Actor Network Theory [27], strategy implementation systems by means of balanced scorecard [25, 26], and results-based monitoring and evaluation in social groups [38].

Because poor understanding of stakeholder's interests often leads to acceptance failure, we must spell out the *value and risks analysis* of the MCR as soon, and as complete as possible, with the means available.

An additional justification for quite detailed value and risk analysis of the MCR services is that network effects complicate the common return on investment and risk analyses within the Method Chunk Repository enabled Community of Practice (MCR COP).

The partial architecture description consists of three parts:

- A *Case Metamodel* supporting the binding of MCR services to interoperability problem owners and objects, and linking prior case-insights into the repository lifecycle.
- An *Enterprise Architecture* emphasizing the value-chains in which the MCR services will develop and grow. Section II.5.2 proposes a set of repository content elements

(these are method chunk products) for the three scales of ICT-reliant work systems: society scale, enterprise (or organisation) scale and person/equipment/tool scale². In line with enterprise architecture insights, a distinction is made between on the one hand the models (data) that describe the enterprise (state-of-affairs) on a continuous basis, and on the other hand the increments to these models (and data processing requirements) that projects built and implement to meet their charters.

- A *Technology Model*, focussing on the technical description of the MCR.

II.5.1 Case Metamodel

In order for MCR to offer services to interoperability problem owners, the MCR should have some knowledge, data and models of the systems where the problems occur. These models themselves are often the products of applying certain method chunks. Certain models must exist before a specific method chunk can be applied. Moreover, both "forward" and reverse (information systems) engineering exist.

To accommodate the wealth of possible method chunks, and support the "any-purpose" "method-chunk-enabled" chaining of "model" products, one arrives at the statement that:

The case metamodel consists of the *product metamodels* of all the method chunks in the MCR, conveniently assembled for the specific problems of the case³.

This statement implies that the case metamodel is dynamic, and will grow with the use of the MCR, as new method chunks are defined.

For a community of interoperable information system practitioners, the introduction of an MCR into the information systems engineering activities represents a major change in those activities. Problem (mess) owners and solvers at three scales are involved: society, businesses and the individual persons [11]. A case can be pertinent to any of these scales, and this scale will influence the method chunks that must be assembled in the process of articulating and solving problems.

Classifying the cases according to scale, the public utility and insurance cases can be situated at the business (inter-organization) scale, and the ATHENA MDI framework at the society scale. Following a situation analysis, the insurance case "project" might built upon the ATHENA MDI framework deliverables.

As was proven in the nineteenth century with the construction of railroads, important social savings can be achieved by solving by capital intensive society scale utility projects, in that case the railroad infrastructure, the mobility needs of innumerable businesses [9].

² On Multi-scale methodology in general, see:

http://is.tm.tue.nl/staff/jgoossenaerts/methodological_reference.htm#H7 (sorry but there are still many dangling references on these pages, i will try to improve on this in november, also pls. note that those pages are not updated according to the contents of this annex).

³ Simplifying, this implies that the metamodel of method chunk presented and illustrated in Annex A serves as a meta-metamodel from the perspective of the case models.

It is now accepted that contemporary businesses have interoperability needs (in addition to the INTEROP literature, see also Gallagher [10]). The inclusion of the Interoperability Service Utility (ISU) Grand Challenge in the recently completed Enterprise Interoperability Research Roadmap [28] proves a growing awareness of the need for utility-style solutions for enterprise interoperability problems. MCR fits in the class of utility-style solutions.

From a Case Metamodel point of view, the addition of two scales (utility at society scale and person scale) affects the case template. The utility services must be defined in such a way that they can contribute to solving the business and person scale interoperability problems, and that they do not magnify the original problem, for instance as a consequence of increased model/data-intensity. The MCR product metamodels are thus seen as key to facilitation of model relationships across scales, in addition to other model relationships across businesses and persons, and across time.

Assuming the presence of a cost-effective ISU, the case interface with MCR is facilitated by case models which are aligned with utility models (society scale) and entity models (person, equipment, tool).

II.5.1.1 Sample Case template

Architectural descriptions can be developed as a set of models. The table below presents a set of models (method chunk products) that could be used to describe a business-scale case. The set of models is based on the COMET methodology [2] which includes support for business architecture description⁴. It is not too hard to map the cases described in this deliverable to the template below.

| Model | Purpose | Usage | Notation | Importance |
|---------------------------------|---|---|--------------------------------|------------|
| Business strategy modelling | Identify stakeholders and business goals, value proposition and risks | End user communication | Informal. Textual and figures. | Essential |
| Business operations modelling | | | | |
| Goal model | | | | Optional |
| Community model | | | | Essential |
| Business process and role model | Identify the business processes | Model-to-model transformation (target = Work Analysis Refinement Model) | UML Activity Model | |
| Business resource model | Identify equipment, information and system resources. | End user communication | UML Class Model | |
| Work analysis refinement model | Refine the Business Process Model. | Model-to-model transformation (target = Component Interaction Model) | UML Activity Model | |

⁴ Referring to Annex A II, this template does not separate the repository content elements with a continuous pertinence for the enterprise, and the increments that are built in projects responding to performance alerts. The situation that fits the use of this template could be described as a greenfield.

| | | | | | |
|-----------------------------|--------------------------------|--|--|--|-----------|
| Requirements modelling | | | | | |
| | Use case model | | | | Essential |
| | System boundary model | Identify system boundaries, the actors and their responsibilities, and the services offered. | Model-to-model transformation (target = Use Case Scenario Model) | UML Use Case Model (optional: UML Sequence Model) | Essential |
| | Use case scenario model | Detailing of the use cases of the System Boundary Model. | Model-to-model transformation (target = Component Interaction Model) | UML Use Case Model | Optional |
| | Non-functional requirements | Identify requirements to the Quality of Service (QoS) such as performance, security level, data quality etc. | End user communication | Textual. (optional: UML Class Model) | Optional |
| | Reference analysis | | | | Optional |
| Component modelling | | | | | |
| | Component structure model | Identify the main system components and their inter-dependencies. | End user communication | UML Class Model | Essential |
| | Component interaction model | Describe the internal behaviour of the offered services. | Model-to-code transformation | UML Activity Model UML Sequence Model | Optional |
| | Component interface model | Identify the interfaces of the services. | Model-to-code transformation | UML Class Model | Essential |
| | Component information model | Identify the structure, relationships and properties of the information items. | Model-to-code transformation | UML Class Model | Essential |
| Platform-specific modelling | | | | | |
| | Platform profile model | A graphical model of the implemented system | Model-to-code transformation | UML Class Model UML Activity Model | Optional |
| | Component implementation model | The running code of the system | Implementation code | Programming code or executable specifications (Java, C#, SQL, BPEL4WS, etc.) | Optional |
| | Deployment model | | | | Optional |

II.5.2 Enterprise Architecture and Project Charters of the MCR

The architecture description (and hence the subdivision of this section) largely follows the multi-scale application of the COMET methodology as described in Annex A1.

We notice that MCR services will be delivered to *social networks* of enterprises and private persons via the web and wireless environment. The high autonomy in such networks poses an MCR adaptation challenge.

Because poor understanding of stakeholder interests often leads to acceptance failure, it is better to spell out the *value and risks analysis* of the MCR at the multiple scales where adoption is anticipated. Network effects complicate the common return on investment and risk analyses within the Method Chunk Repository enabled Community of Practice (MCR COP). Due to time limits we will only draw up the broad lines of the repository content elements for MCR and the related project charters.

At each scale we address both the architecture descriptions that belong in repositories and the project charter to which the MCR would offer a solution.

II.5.2.1 Society Scale Architecture Descriptions and Project Charter

The MCR must be developed in a socio-economic environment in which there are no clear leaders in terms of power to coerce all stakeholders to adopt and start using specific methodologies nor MCR. It is known that in a context lacking clear leadership, the decision making is slow.

The combined effects of socio-diversity and techno-diversity [11] cause an exploding and asset-threatening interoperability problem. Simultaneously, a growing fragmentation has been witnessed as traditional knowledge accumulation practices fail to cope with the hyper-competitive, dynamic and high-frequency disruptive technology cycles. In an effort to cope with these new trends, INTEROP NOE was created in FP 6 with the support of the European Commission. The growing complexity of the management and engineering of contemporary socio-technical systems such as ICT-reliant enterprises, and the rising number of artefacts involved in these reflective activities have convinced TG 6 Method Engineering to investigate and prototype-develop a repository of method chunks or user-configured and user-composed aggregated methods (INTEROP method repository) supporting different interoperability issues in the domains of Enterprise Modelling (EM), Ontology (ONT) and Architecture & Platforms (A&P).

These and other considerations must be reflected in the society scale Methodology/MCR VARM and WOM's (

Table 2), prior to the realization of the MCR components (see COMET for more details) and their implementation at the adopting population.

Table 2. Repository Content Elements (VARM and WOM) for Methodology Deployment

| Repository Content Element | Shorthand | Description |
|------------------------------|---------------|---|
| Mission Statement | MCR.VARM.MS | To achieve and sustain MCR services for Interoperability problem owners, their service providers and their knowledge suppliers. E.g., in the society: <i>"substitute significant amount of ad-hoc methodology decisions by method chunk assemblies, and owe a much improved value and risk profile to it."</i> (e.g. avoid cost overruns, project failure, rapid obsolescence,...) |
| Domain Statement | MCR.VARM.DS | <p>The stakeholder roles in the MCR collaborations include:</p> <ul style="list-style-type: none"> • MCR Owner/provider • The MC experts • The MC end-users (businesses with interoperability problems) • The ESA solution providers and enterprise software vendors • The standards and regulatory bodies • The knowledge institutes • The publishers of method chunks and cases demonstrating their application <p>The objects involved in these collaborations include those in the extension of the metamodel of the method chunks, as well as those in the extension of the method chunk product metamodels, .</p> |
| Market Indicator Register | MCR.VARM.MIR | A range of indicators that the MCR owners and other stakeholders can use to verify the extend of MCR service use to solve interoperability problems (nr. of users,...). |
| Market Value & Risk Register | MCR.VARM.MVRR | <p>Relevant value exchanges include:</p> <ul style="list-style-type: none"> • the input of MC knowledge by the expert into the MCR, the return in exchange of some recognition (reputation); • the exchange (for money) of interoperability-problem-solving MC services by the MC end-users; • ... <p>Relevant risks for these value exchanges include:</p> <ul style="list-style-type: none"> • the non-applicability of method chunks included in the MCR • the lack of ability to match chunk theft of cash held by any party in the eco-system, • the lack of chunks to complete the method chain for a (customer) problem owner. |
| Principal Model | MCR.WOM.PM | The principals that are recognized by the MCR include private persons, businesses, academic institutions, specific public agencies. Instances of the classes in the Principal model will be bound to the stakeholder roles defined in the domain statement. |
| Market Asset Model | MCR.WOM.MAM | A model of all the MCR assets: it will include method chunks, their utilization intensity, cases to which they have been applied, contracts with specific principals, endowing these with the rights, restrictions and responsibilities connected to specific stakeholder roles w.r.t. specific assets. |
| Market Resource | MCR.WOM.MRM | A model of all the MCR resources that are created to support the MCR-enabled information systems engineering, the method chunk life cycle |

| | | |
|------------------------------|-------------|--|
| Model | | management and the interfaces to the different stakeholder roles. |
| Interaction Refinement Model | MCR.WOM.IRM | A model of all the principal interactions that the MCR architects have specified (in contracts) to ensure the proper and successful use of MCR services. |

Applying the project descriptors listed in Annex A1, the project charter for MCR introduction into the methodology deployment by the "IS-reliant social networks" is drafted in the table below.

Table 3. Incepting MCR in a society of Methodology Ad-Hocracy

| | |
|-------------------|---|
| Problem Statement | See the table Problems, Opportunities and Directives for Method Chunk Repository Services in D1. |
| Assumptions | A sufficient number of method chunks is known. |
| Vision Statement | Method chunks sourced from multiple MCR's will be assembled in order to resolve interoperability problems in specific situations. |
| Scoping Statement | Restriction to Interoperability problems. |
| Goal Model | Omitted for now. |
| Project Risks | Omitted for now. |

II.5.2.2 Organization-scale Architecture Descriptions and Project Charter

Elaboration of the Business-scale (organization) repository content elements starts from the role models that were identified at the society scale. For the MCR collaborations the important stakeholder roles include:

- MCR Owner/provider
- The MC end-users (businesses with interoperability problems):
- The ESA solution providers and enterprise software vendors
- The standards and regulatory bodies
- The knowledge institutes
- The publishers.

The MC end-users role is here used to illustrate the manner of refining stakeholder interfaces (roles) with respect to the MCR collaborations. Where interoperability problems lead to enterprise asset erosion, the first kind of stakeholders for which an MCR can create value are the *enterprise owners*. Periodically, enterprise owners decide to further develop the enterprise (an *IS reliant work system*) that is creating value for them. In each and every enterprise, such a decision can be due to a performance problem that is observed during the (scorecard based) monitoring and evaluation of the primary processes and the assets sustaining these processes.

Within the development life cycle stage of the enterprise, the performance alert is scoped, and the causes for the problem are identified (problem analysis). Interoperability problems are created or resolved in the development responses that affect the IS reliant work system. To be useful to the enterprise, it must be "easy" (and cost-effective at least) for the enterprise stakeholder roles to involve the MCR services. This implies interface-customisation demands (role-bindings) for the interactions defined in MCR.

The table below illustrates the state-of-affairs enterprise architecture descriptions that must be faced as an enterprise decides to link-in itself into the MCR user community. The table with the project charter illustrates issues as one proceeds in identifying the change needs for the enterprise. The "link-into-the-MCR-user-community" project must deliver so-called "enterprise-architecture-slices" that must be added to the MC end-user repository, and must be implemented in the enterprise ICT project practices.

Table 4. Repository Content Elements (VARM and WOM) for MC end-user (business)

| Repository Content Element | Shorthand | Description |
|------------------------------------|---------------|---|
| Organization Mission Statement | ORM.VARM.OMS | The Mission Statement of the Enterprise considered. |
| Organization Context Statement | ORM.VARM.ECS | The identification of the different kinds of principals and objects that the Enterprise engages with in order to fulfil its missions. These include characterization of the market segments, the products and services provided etc. |
| Organization Indicator Register | ORM.VARM.OIR | A register with the names, definitions and specifications of the indicators that the Organization uses to verify the extend of achievement regarding the service and product flow the Organization wants to offer to the market, and regarding the Organization's compliance with the society's enacted institutions. A large number of enterprises use Balanced Scorecard. |
| Organization Value & Risk Register | ORM.VARM.OVRR | A register with the identified sources of values and risks, their indicators, references to their (past) measurements and future expectations. These values and risks influence the chances of achieving the service and product flow the Organization wants to offer to the market, as well as its compliance with the society's enacted institutions. |
| Community Model | ORM.WOM.CM | A model that includes all the principals that are recognized by the Organization's work processes and their refinements (as specified in ORM.WOM.WRM. It must be aligned with SRM.WOM.PM, but typically it will include specialization classes to differentiate principals in relation to ORM.VARM.OMS. |
| Organization Asset Model | ORM.WOM.OAM | A model of all the assets that the Organization controls in order to sustain the continuous service flow it wants to offer to its customers, in exchange for their attention or money. It will include facilities, brands, distribution channels, equipment, intellectual property rights, etc. |
| Organization Resource Model | ORM.WOM.ORM | A model of all the resources that the Organization creates, acquires and maintains to realize the flow of products and services identified in its mission statement. It will cover the information flows, organisational |

| | | |
|---|---|---|
|  | <p style="text-align: center;">INTEROP Interoperability Research for Networked Enterprises Applications and Software</p> | <p style="text-align: center;">IST-508011  Information Society</p> |
|---|---|---|

| | | |
|-----------------------|-------------|---|
| | | structures, worker-roles, etc. |
| Work Refinement Model | ORM.WOM.WRM | A model of all the Organization processes and SRM.WOM.IRM interaction specializations that the Organization has specified to fulfil its missions (ORM.VARM.OMS), pursue its values while avoiding the risks (ORM.VARM.MVRR). As far as interactions and the use of restricted assets and resources is concerned, the models must be compliant with SRM.WOM.IRM. (e.g. as in the ISO 9000 certification documents) |

Table 5: Project Charter for Incepting MCR in a end-user Enterprise suffering from Methodology Ad-Hocracy

| | |
|--------------------------|--|
| <i>Problem Statement</i> | Interoperability problems contribute to low project predictability, frequent cost over-runs and failed projects, |
| <i>Assumptions</i> | (Experts in) the enterprise have a fair understanding/ awareness of its situation and see MCR as a means to do something about it. |
| <i>Vision Statement</i> | The enterprise will use MCR assembly services; it will provide situational method chunks sourced from multiple MCR's will be assembled in order to resolve interoperability problems in specific situations. |
| <i>Scoping Statement</i> | Restriction to Interoperability problems (as yet). |
| <i>Goal Model</i> | Much better control of Business/IT aligned projects. |
| <i>Project Risks</i> | Stranding (e.g. due to MCR services becoming obsolete as superior solutions to interoperability problems reaches the market). |

Execution of the MCR Inception project at the end-user enterprise will influence the work methods and resource use of the team members. These changes can be expressed as additions or modifications to OAM, ORM and WRM of ORM.WOM.

II.5.2.3 Person-scale Architecture Descriptions and Learning Targets

Person-scale repository content elements must be elaborated for the stakeholder roles:

- MC and interoperability experts
- MC end-users (the persons who must solve interoperability problems)
- trainees
- educators
- the roles of the business stakeholders that must interact with the MCR

Equipment and tools are operating at the same scale as persons. Repository content elements must also be elaborated for them. This is not addressed here.

Table 6: Repository Content Elements (VARM and WOM) for MC end-user (person)

| Repository Content Element | Shorthand | Description |
|----------------------------|---------------|---|
| Ambition Model | ERM.VARM.AM | A description of the general ambitions of a person, including the targets for the roles he or she fills professionally. Compliance with Society-enacted institutions is a constraint for AM. |
| Person Context Statement | ERM.VARM.PCS | The identification of the different kinds and instances of society members that the person engages with in order to achieve its ambitions. (labour contracts, learning contracts, customer relations filled) |
| Person Indicator Register | ERM.VARM.EIR | A register with the names, definitions and specifications of the indicators that the person uses to verify the extend of achieving his/her ambitions and targets. |
| Ens Value & Risk Register | ERM.VARM.EVRR | A register with the identified sources of values and risks that influence a person/equipment/tool's ability and chances to achieve its ambitions and targets. Poor methodology sharing is one such risk driver (vulnerability). |
| Role Model | ERM.WOM.RM | A model that includes all the principals that the person engages with during his/her ICT project work. |
| Ens Asset Model | ERM.WOM.EAM | A model of all the assets that the person owns in order to achieve his or her ambitions. |
| Ens Resource Model | ERM.WOM.ERM | A model of all the resources that the person uses to protect and sustain his/her assets and use them in achieving ambitions. |
| Behaviour Model | ERM.WOM.BM | A model of all the person processes and interaction specializations that the person has selected to achieve his/her ambitions and targets: to pursue values while avoiding risks. |

II.5.3 Technical Architecture of the MCR

This section describes the architecture for a collaborative method engineering platform (CMEP) focusing on the role of the MCR.

II.5.3.1 Use case descriptions

A collaborative platform for situational method engineering must support two main activities: situation-specific method construction and method application in the corresponding system development project. The method construction activity requires capabilities for reusable method chunks definition, storage and classification with respect to the problems they help to solve. It also aims to support the characterisation of each project situation and selection and assembly of method chunks fitting the situation at hand. The method application requires services for the obtained method enactment and evaluation of its applicability in the corresponding situation. Figure 10 summaries this in the form of a UML use case diagram that depicts the main actors and their usage of the platform.

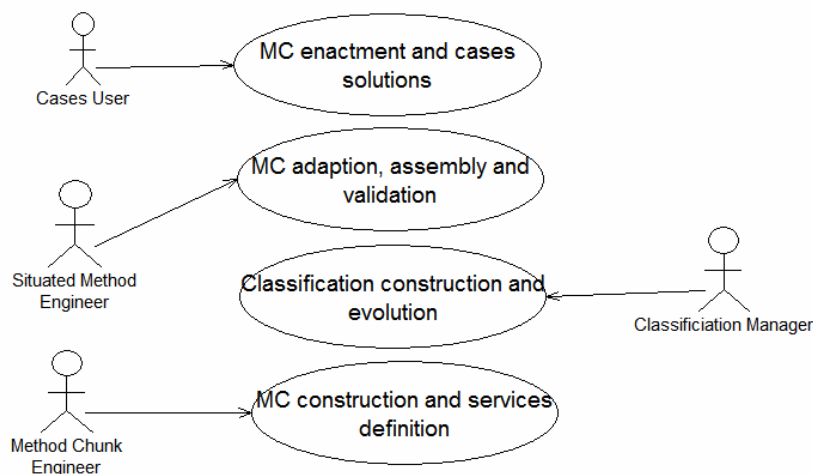


Figure 10. System boundary model

Use cases are part of a structured approach to capture all interactions a system should do and at the same time define all interacting user groups and systems. The use cases describe how actors interact with the system. Table 7 gives a description of the actor's goals. The main use cases help us to identify services that the MCR has to provide to the end-users.

Table 7. Description of the Actors

| Actor | Description |
|--------------------------|--|
| Cases user | <p>The goal of the cases user is to easily and efficiently be able to search for and test/analyse/apply method chunks to specific cases, as well as describe experience of using these method chunks in his/her specific case.</p> <p>The cases user will apply the case-specific method in the development of a corresponding project and will provide an experience report including the evaluation of the applied method chunks and their fitness to this case.</p> |
| Situated method engineer | <p>The goal of the situated method engineer is to find a set of method chunks that can be assembled into a coherent method that addressing a particular (interoperability) development/analysis need.</p> <p>The situated method engineer is in charge of constructing a case-specific method for each case. His/her work consists of three main tasks: characterising the case situation by using the classification framework, selecting method chunks satisfying this situation and assembling retrieved method chunks in order to provide a coherent and complete method for the specific case</p> |
| Method chunk engineer | <p>The goal of the method chunk engineer is to capture knowledge to specific problems as reusable method chunks that can be used in different situations.</p> <p>The method chunk engineer is an expert in the method engineering domain. His/her role is to populate the MCR with method chunks, which can be extracted from existing traditional methods or defined from scratch on the basis of domain knowledge and experience. The method chunk engineer will also develop services for method chunks application and provide a descriptor for each method chunk characterising, with the help of the</p> |

| | |
|------------------------|--|
| | classification framework, the context of its application and the interoperability issues it helps to solve. |
| Classification manager | <p>The goal of the classification manager is to develop and evolve classification schemes for classifying method chunks so that they are easy to search and navigate.</p> <p>The classification manager is responsible for defining and managing the method chunk classification framework. Such a framework should be extensible and evolutionary. Good knowledge about the information systems development domain and some selected application or problem domain, such as interoperability in our case, is required to enact this role.</p> |

II.5.3.2 Service architecture

The use cases serve as a starting point for more detailed scenario descriptions where one can describe the human-computer interaction and working environment of the end-users. A high-level description (Figure 11) of the MCR can be derived from the use cases.

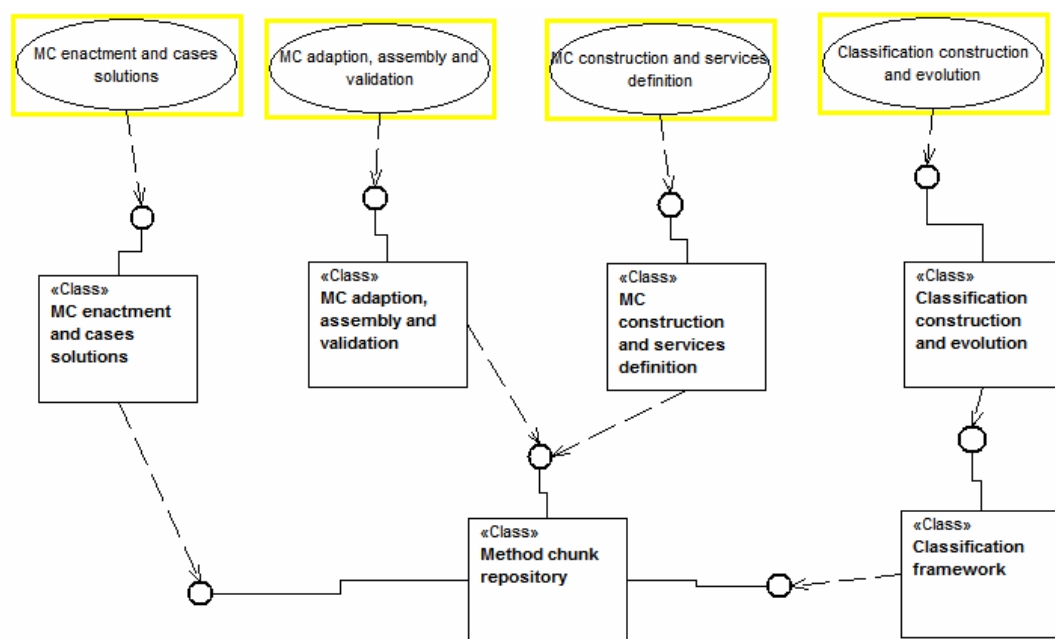


Figure 11: High-level service architecture for collaborative method engineering platform (CMEP)

A first iteration of the architecture implies that each use case will be supported by a separate service as shown in Figure 11. However, when refining the use case (detailing the scenarios) you may find that a use case can be broken down into smaller separate user tasks which each require their own services.

II.5.3.3 Provided services

Based on the analysis of the use case scenarios the following services are identified:

Table 8. Provide services

| Service | Description |
|--|--|
| Metamodelling facility | Develop and maintain metamodels that describe the product part of the method chunk. |
| Search facility | Services to search/browse method chunks according to particular needs. |
| Method chunk authoring | Services to define method chunks according to the method chunk metamodel which includes: <ul style="list-style-type: none"> • Graphical editor (model template) for composing your own method chunk • Define roles, work products, tasks, guidance • Define templates, guidelines, examples, checklists • Classify the method chunk |
| Process authoring (composer) | Services to assemble situational methods from method chunks which includes: <ul style="list-style-type: none"> • Graphical editor (model template) for method assembly • Define workflows and work breakdown structures • Software development process patterns • Define reusable development processes |
| Cases facility | Services to describe cases and capture experiences with particular method chunks. |
| Method chunk repository | <p>The method chunk repository provides library services for storing and updating two types of interconnected knowledge</p> <ul style="list-style-type: none"> • the method knowledge expressed in the form of reusable method chunks; and • the knowledge related to the experience of method chunks application in specific industrial cases <p>The repository must also provide versioning capabilities and define a standard/common interchange format for the exchange (storage and retrieval) of method chunks and experience reports.</p> |
| Classification manager | Services for classifying method chunks. In order to match the problem situation of a particular case to method chunks thus enabling a solution, we need a mechanism supporting method chunks indexation on the one hand and situation assessment on the other hand. This mechanism is referred to as a matching/classification framework |
| Link to enactment of the method or project management system | Services to enact situational methods in specific projects. The services should support retrieval and assembly of method chunks to form a situational method. Retrieval involves finding and selecting method chunks that suits the specific project and can utilize the project characterization to match against classifications of method chunks. During assembly the tool should allow a flexible way to manipulate chunks within the method under construction. The origin for this requirement is [14]. |

II.5.4 Implementation description

In the case of TG6 a number of *prototype implementations* of the services identified above have been implemented in the Metis. The first implementation is reported in section II.4. It faithfully represents the method chunk metamodel as outlined in this deliverable. The second is reported in appendix A4. It is an alternative view that takes the metamodel from Brian

Henderson-Sellers as a guideline. Details can be found in appendix A4. A third prototype implementation based on the ConceptBase meta database system. Its definition is contained in appendix A5.

II.6 Conclusion

The deliverable DTG6.2 has the purpose to finalize the proposed structure of the method chunk repository (MCR). The final structure is reported in appendix 3 (also a paper accepted by the prestigious ER conference) augmented by the additions of section II.4 for classifying interoperability problems. The final structure is based on real case studies undertaken by task group TG6 to elicitate the requirements for a method chunk repository. The findings of these case studies are in line with the findings of the INTEROP Domain Interoperability. Since we have the goal to validate our metamodel by a prototypical implementation, our proposed structure is more detailed and also more formal in the sense that it is the basis for developing a method chunk repository for interoperability.

We also derived text-based template for entering method chunks and cases into the MCR. The purpose of this template is to facilitate acquisition of method chunks. Furthermore, we made an analysis of the architecture of a full-fledged MCR using the COMET approach as general guideline. This approach focuses on identifying different user roles such as method chunk engineer, case engineer etc.

Finally, we made in total three prototypical implementations. The first uses the METIS tool to represent the MCR metamodel as outlined in this deliverable. Second, the same Metis tool was used to realize a competing metamodel taken from proposal by Brian Henderson-Sellers. This metamodel does not focus on interoperability but it re-affirmed the usability of the METIS tool for creating a MCR. Finally, our MCR metamodel was prototypically implemented by the ConceptBase meta database system to conform that the metamodel is not limited to METIS as implementation platform.

The joint work on the deliverable and the prototypical implementation has led to a high degree of collaboration inside the task group. We feel encouraged by the results to continue our work towards a method chunk repository. Our experience with the prototypes shows that our ideas are actually implementable though a lot of work would have to be done to produce a stable and full-fledged MCR. While a full-fledged implementation is beyond the scope and goal of the task group, we have paved the way to produce such a system.

In the remaining time of the INTEROP Network we plan to work on a tutorial about the MCR and publish the final results at a highly visible place.

II.7 References

1. AOS, "JACK Intelligent Agents", The Agent Oriented Software Group (AOS). <http://www.agent-software.com/shared/home/> (accessed: 2006).
2. ATHENA, "ATHENA Public Web Site", ATHENA Integrated Project (IST-507849). <http://www.athena-ip.org/> (accessed: 2005).

3. Backlund P., Ralyté J., Jeusfeld M., Kühn H., Arni-Bloch N., Goossenaerts J.B.M. and Lillehagen F. (2006). An Interoperability Classification Framework for Method Chunk Repositories. Proceedings of the 15th International Conference on Information Systems Development (ISD'2006), Budapest, Hungary, 31 August - 2 September 2006, Springer Verlag.
4. Berre, A.-J., B. Elvesæter, J. Ø. Agedal, J. Oldevik, A. Solberg, B. Nordmoen (2004) COMET – Component and Model-based development Methodology; Methodology Handbook, SINTEF ICT, Norway.
5. Booth, D., H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, and D. Orchard, "Web Services Architecture", World Wide Web Consortium (W3C), W3C Working Group Note, 11 February 2004. <http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/>
6. Brinkkemper S., Saeki, M. and Harmsen, F. (1998). Assembly Techniques for Method Engineering. 10th Conference on Advanced Information Systems Engineering, CAiSE'98. Springer, LNCS 1413, pp.381-400.
7. Chen D. (Ed.) (2005) Practices, principles and patterns for interoperability. Deliverable D6.1 Interop Network of Excellence IST – 508011. <http://interop-noe.org/deliv/d6.1/> Accessed 2006-11-19.
8. Eclipse.org, "Eclipse Process Framework Project (EPF)". <http://www.eclipse.org/epf/> (accessed: 2006).
9. Fogel, R. W. (1979) Notes on the Social Saving Controversy. The Journal of Economic History. Vol. 39 (1) pp 1-54.
10. Gallagher, M. P., O'Conner, A. C., Dettbarn, J. L. Jr., Gilding, A., & Gilday, L. T. (2004). Cost Analysis of Inadequate Interoperability in the U.S. Capital Facilities Industry. NIST GCR 04-867
11. Goossenaerts, J.B.M. (2004) Interoperability in the Model Accelerated Society (2004) in: P. Cunningham and M. Cunningham (2004) eAdoption and the Knowledge Economy: Issues, Applications, Case Studies. IOS Press Amsterdam, pages 225-232.
12. Goossenaerts, J.B.M. (2006) Monitoring, Evaluation & Decision Interoperability in Networked Organisations. In EMOI 2006 Proceedings.
13. Graham, I., Henderson-Sellers, B. and Younessi, H., 1997, The OPEN Process Specification. Addison-Wesley, Harlow, UK.
14. Harmsen, A.F. (1997). Situational Method Engineering. Moret Ernst & Young.
15. Iivari, J., Hirschheim, R. and Klein, H.K (2004) Towards a distinctive body of knowledge for Information Systems experts: coding ISD process knowledge in two IS journals. Information Systems Journal (14) pp. 313-342.
16. Object Management Group, "OMG Model Driven Architecture", Object Management Group (OMG), 2005. <http://www.omg.org/mda/>
17. OMG, "MDA Guide Version 1.0.1", Object Management Group (OMG), Document omg/03-06-01, June 2003. <http://www.omg.org/docs/omg/03-06-01.pdf>
18. OMG, "UML 2.0 Infrastructure Specification", Object Management Group (OMG), Document ptc/03-09-15, December 2003. <http://www.omg.org/docs/ptc/03-09-15.pdf>
19. OMG, "UML 2.0 Superstructure Specification", Object Management Group (OMG), Document ptc/03-08-02, August 2003. <http://www.omg.org/docs/ptc/03-08-02.pdf>
20. OMG, "Meta Object Facility (MOF) 2.0 Core Specification", Object Management Group (OMG), Document ptc/04-10-15, October 2004. <http://www.omg.org/docs/ptc/04-10-15.pdf>
21. OMG, "MOF 2.0/XMI Mapping Specification, v2.1", Object Management Group (OMG), Document formal/05-09-01, September 2005. <http://www.omg.org/docs/formal/05-09-01.pdf>
22. OMG, "Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification", Object Management Group (OMG), Document ptc/05-11-01, November 2005. <http://www.omg.org/docs/ptc/05-11-01.pdf>
23. OMG, "Software Process Engineering Metamodel Specification, Version 1.1", Object Management Group (OMG), Document formal/05-01-06, January 2005. <http://www.omg.org/docs/formal/05-01-06.pdf>

24. OMG, "Common Warehouse Metamodel (CWM), Version 1.1", Object Management Group (OMG), Document formal/03-03-02, 2003. <http://www.omg.org/docs/formal/03-03-02.pdf>
25. Kaplan. R.S., D. P. Norton (1993) The Balanced Scorecard - Measures that Drive Performance. Harvard Business Review, January/February 1993.
26. Kaplan. R.S., D. P. Norton (1996) Using the Balanced Scorecard as a Strategic Management System. Harvard Business Review, January/February 1996.
27. Latour, B. (1987) Science in Action: How to Follow Scientists and Engineers through Society. Harvard University Press, Cambridge MA.
28. Li, M., Cabral, R., Doumeingts, G. & Popplewell, K. (2006) *Enterprise Interoperability Research Roadmap*. European Commission, ftp://ftp.cordis.europa.eu/pub/ist/docs/directorate_d/ebusiness/ei-roadmap-final_en.pdf
29. modelbased.net, "ATHENA Model-Driven Interoperability (MDI) Framework". <http://modelbased.net/mdi/> (accessed: 2006).
30. modelbased.net, "COMET - Component and model-based development methodology". <http://modelbased.net/comet/> (accessed: 2006).
31. modelbased.net, "ATHENA Service-Oriented Interoperability (SOI) Framework". <http://modelbased.net/soi/> (accessed: 2006).
32. Mirbel I. and Ralyté J. (2006) Situational Method Engineering: Combining Assembly-Based and Roadmap-Driven Approaches, Requirements Engineering, 11(1), pp. 58–78.
33. Ralyté J., Backlund P., Kühn H., Jeusfeld M. (2006). Method Chunks for Interoperability. Proceedings of the 25th International Conference on Conceptual Modelling (ER'2006). Tucson, Arizona, USA, November 6-9 2006.
34. Ralyté J. and Rolland C. (2001). An Approach for Method Reengineering. Proceedings of the 20th International Conference on Conceptual Modeling (ER2001), LNCS 2224, Springer-Verlag, pp.471-484.
35. Ralyté, J. (2004). Towards Situational Methods for Information Systems Development: Engineering Reusable Method Chunks. Proceedings of the International Conference on Information Systems Development (ISD'04), Vilnius, Lithuania, September 2004.
36. SWEBOK (2004) Guide to the Software Engineering Body of Knowledge 2004 Version. Available at <http://www.swebok.org/>. Accessed 2005-12-08
37. Troux Technologies (2006) <http://www.troux.com/products/metis/>, Metis by Troux. Online. March 30, 2006.
38. Zall Kusek, J., R.C. Rist (2004) Ten Steps to a Results-Based Monitoring and Evaluation System; A Handbook for Development Practitioners, The International Bank for Reconstruction and Development / The World Bank, Washington D.C.

PART III : Appendices

The following appendices accompany the deliverable:

Appendix A1: Jan Goossenaerts (2006): Definition of the COMET methodology for MCR detailed design

Appendix A2: Backlund, P. et.al. (2006): An Interoperability Classification Framework for Method Chunk Repositories. 15th Intl. Conference in Information Systems Development, Budapest, Hungary, August 31 to September 2, 2006.

Appendix A3: Ralyté, J., P. Backlund, H. Kühn, and M.A. Jeusfeld (2006): Method chunks for interoperability. Draft, to appear in Proceedings 25th International Conference on Conceptual Modeling (ER-2006), Tucson, Az., USA, Nov. 6-9, 2006. **Appendix A4:** Alternative Metis-based specification of a MCR

Appendix A4: Roland Norberg (2006): Representing Methodological Knowledge - Metamodeling for a method chunk repository. Master dissertation, Högskolan Skövde, Sweden, 2006.

Appendix A5: Manfred Jeusfeld (2006): ConceptBase-based specification of a MCR, TG6 internal working document, Tilburg University, 2006.

Appendix A6: Brian Elvesæter (2006): Model-driven development case – ATHENA model-driven interoperability (MDI) framework, TG6 internal working document, SINTEF ICT, 2006.