Deliverable DTG6.3

Tutorial on Situational Method Engineering for Interoperability

<table>
<thead>
<tr>
<th>Deliverable No. and title:</th>
<th>DTG6.3 Tutorial on Situational Method Engineering for Interoperability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification:</td>
<td>Confidential</td>
</tr>
<tr>
<td>Project Responsible:</td>
<td>Jolita Ralyté, UNIGE #52</td>
</tr>
<tr>
<td>Main author:</td>
<td>Jolita Ralyté, UNIGE #52</td>
</tr>
<tr>
<td>Contributing authors:</td>
<td>Per Backlund, HS, #48</td>
</tr>
<tr>
<td></td>
<td>Manfred Jeusfeld, UNTILB, #44</td>
</tr>
<tr>
<td></td>
<td>Harald Kühn, BOC, #28</td>
</tr>
<tr>
<td></td>
<td>Jan Goossenaerts, TU Eindhoven, #14</td>
</tr>
<tr>
<td></td>
<td>Nicolas Arni-Bloch, UNIGE #52</td>
</tr>
<tr>
<td></td>
<td>Brian Elvesæter, SINTEF, #16</td>
</tr>
<tr>
<td></td>
<td>Frank Lillehagen, Troux, #55</td>
</tr>
<tr>
<td>Completion Date:</td>
<td>2007-05-07</td>
</tr>
<tr>
<td>Version</td>
<td>Version 0.2</td>
</tr>
</tbody>
</table>
# Table of contents

Deliverable DTG6.3 .......................................................................................................................... 1
Executive Summary ......................................................................................................................... 3
PART I ............................................................................................................................................... 4
  I.1 Introduction.............................................................................................................................. 4
  I.2 Methods to produce the deliverable ....................................................................................... 4
  I.3 Main Results since the last deliverable ................................................................................. 4
  I.4 Conclusion .............................................................................................................................. 5
PART II .............................................................................................................................................. 6
PART III : Annexes .......................................................................................................................... 7
Executive Summary

This deliverable summarises the achievements of the TG6 in the form of tutorial entitled Situational Method Engineering for Interoperability.

The tutorial starts with the introduction to situational method engineering and the motivation how this method engineering discipline could support enterprise systems interoperability projects. It includes the definition of a method chunk, the template for method chunks construction and examples of method chunks.

Then, the notion of a method chunk repository (MCR) is defined and motivated. The main objective of such a repository is to support capitalisation and reuse of the methodological knowledge related to the interoperability domain. This knowledge can be extracted from existing methods and technique or captured from best practices and experience in realising various interoperability projects. Besides method chunks, the MCR also aims to capture knowledge about the application of these chunks in different projects and the evaluation of their fitness to these projects. The MCR became even more interesting if it is integrated into a collaborative method engineering tool (meta-case tool). In this tutorial we design the architecture of such a tool. We specify the actors interacting with the MCR and the services to be provided by the tool.

Finally, the interoperability classification framework is proposed as an indexation mechanism to be used with the MCR. This framework aims to support method chunk characterisation in order to link method chunks to the interoperability problems they address. It also supports situation assessment of an application case (project) in order to identify the interoperability problems that occur in this case. Finally, this framework is a basis for the matching mechanism between the case situation and the method chunks.

The last results of the TG6 have been published in the proceedings I-ESA’07. This paper has won the Best Paper Award; a copy is included in the appendix.
PART I

I.1 Introduction

The TG6 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 different application 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 has analyzed realistic cases to extract from them the definite structure of the method chunk repository, called method chunk metamodel. The architecture of a method engineering tool allowing this repository to be functional was designed. Finally, 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 The last period, from M36 to M42, was dedicated to the validation and exploitation of the result reported in the deliverables DTG6.1 and DTG6.2 and their publication in a conference paper. Therefore, the deliverable DTG6.3 is a tutorial which integrates all the work done by the TG6 and can be used to disseminate the results of the task group.

I.2 Methods to produce the deliverable

According to the roadmap of the TG6 and its extension up to M42, two main tasks have been done during the last six months period: writing a joint conference paper and elaboration of a tutorial. The tutorial was built on the base of the two previous deliverables and the joint papers. A two days meeting has been organised in Geneva in the beginning of February to define the plan and content of the tutorial.

The joint paper has been initiated during INTEROP meeting. Elaboration and finalisation have been done during several Skype conferences. 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.

I.3 Main Results since the last deliverable

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

1. A tutorial “Situational Method Engineering for Interoperability” including all main achievements of the TG6.

2. A third co-authored paper published in the proceedings of the 3rd International Conference I-ESA’07. This paper won the Best Paper Award.
I.4 Conclusion

We consider that the TG6 has met its objectives (see the TG6 activity report) and has produced a tutorial accessible, under the creative common licence, for the WP10 and the future V-Lab.
PART II

This deliverable summarises the main TG6 achievements already presented in the deliverables DTG6.1 and DTG6.2 and three co-authored publications (published in the proceedings of international conferences ISD’06, ER’06 and I-ESA’07) in the form of tutorial entitled Situational Method Engineering for Interoperability.

The tutorial is included in the Annex A1 of this document. It has three main sections, introduction and conclusion.

The introduction part of the tutorial reports TG6 objectives and motivations. It provides interoperability definitions and a frame to understand and characterise the domain of interoperability in context of two or more enterprises.

The first main section of the tutorial defines the notion of Situational Method Engineering and explains how this Method Engineering discipline could support enterprise systems interoperability projects. The notion of method chunk is defined in this section as well as a template for method chunk construction and examples of method chunks are provided. Each method chunk deals with some particular interoperability problem.

The second main part of the tutorial represents the notion of a method chunk repository (MCR). The main objective of such a repository is to support capitalisation and reuse of the methodological knowledge related to the interoperability domain. This knowledge can be extracted from existing methods and technique or captured from best practices and experience in realising various interoperability projects. We claim that it is impossible to create one unified method supporting all interoperability problems but rather a project-specific method can be created by assembling several method chunks stored in the repository according to the problems identified in this particular application case.

Besides method chunks, the MCR also aims to capture knowledge about the application of the chunks in different projects and the evaluation of their fitness to these projects. The MCR became even more interesting if it is used with a collaborative method engineering tool (meta-case tool). In this tutorial we design the architecture of such a tool. We identify the main actors interacting with the MCR and the services to be provided by the tool.

The third part of the deliverable proposes an interoperability classification framework. This framework should serve as indexation mechanism related to the MCR. This framework aims to support method chunks characterisation in order to link them to the interoperability problems that they address. Moreover, it also aims to support project situation assessment in order to identify the interoperability problems that occur in this application case. Finally, this framework allows to identify and to select method chunks which correspond to the application case situation.

The last results of the TG6 have been published in the proceedings I-ESA’07. This paper has won the Best Paper Award; a copy is included in the Annex A2.
PART III : Annexes

Annexe A1: TG6 Tutorial “Situational Method Engineering for Interoperability”

Annexe A2: Co-authored publication:

Situational Method Engineering for Interoperability

Jolita Ralytė
University of Geneva

Contributors NoE Interop TG6:
Per Backlund (HS #48), Manfred A. Jeusfeld (UNTILB, #44), Harald Kühn (BOC #28), Jan Goossenaerts (TUE, #14), Nicolas Arni-Bloch (UNIGE #52), Brian Elvesæter (SINTEF #16), Frank Lillehagen (Troux, #55)

Plan of the Tutorial

- Introduction
- Situational method engineering to support interoperability
- Method Chunks Repository for Interoperability
- Interoperability classification framework
- Conclusion
Plan of the Tutorial

Introduction
- Definitions
- Characterising the Interoperability domain
- Research context – Interoperability is Knowledge
- Research objectives
- Situational method engineering to support interoperability
- Method Chunk Repository for Interoperability
- Interoperability classification framework
- Conclusion

Interoperability: Definition

Capability of a system or a product to work with other systems or products without specific effort from the user

Capacity of an enterprise software or application to interact with others
Characterising the Interoperability Domain

Interoperability is Knowledge!

- There is no general method to achieve interoperability
- Different situations require adapted solutions
- What works in one project may fail in another
- The human factor is decisive
- Interoperability is emerging from best practices
- The complexity of the socio-technical system prevents us from understanding it sufficiently to develop a theory of interoperability
**Research Objectives**

Situational Method Engineering as a means for encoding situated knowledge about achieving interoperability in the form of method chunks

- Reusable method chunks – each dealing with some particular interoperability problem
- Repository of Method Chunks
- Method Engineering Platform for Situational Method Engineering

**Plan of the Tutorial**

- Introduction
- Situational method engineering to support interoperability:
  - Introduction to Situational Method Engineering (SME)
  - Method chunk
  - Guidelines for method chunks definition
- Method Chunk Repository for Interoperability
- Interoperability classification framework
- Conclusion
Situational Method Engineering (SME)

Situational method engineering is the discipline to build project-specific methods, called situational methods, from parts of the existing methods, called method fragments

[Brinkkemper 96]

Objective: ‘On the fly’ method construction / adaptation / extension to support project specific method situation

Propositions:
- Selection and assembly of method fragments / components / chunks
- Extension and enhancement of existing methods by new functionalities, ways of working, properties etc.
- Requirements for a CAME tool supporting SME

Overview of the Approach

I. Reengineering of methods into method chunks & Construction of new method chunks

II. Assembly-based Situation-specific Method Construction

[Ralyté & Rolland, 2001]
**From Method Meta-model to Method Chunk**

- **Product Model**: a set of concepts and constraints to describe a system at different levels of abstraction and rules to use them.
- **Process Model**: a way of working, a dynamic perspective of the method. It describes how to construct the corresponding product model. The product is a result of the process model application.

**Method Chunk**

- **Chunk Descriptor**
  - **Interface**: <Situation, Intention>
  - **Body**
    - **Product Part**
    - **Process Part**
  - **Conditions of application**
  - **Objective**
  - **Guidelines to help the engineer to apply the chunk**
  - **Reuse conditions in the ISD process**
  - **Interoperability problem**

- **Panel**
  - A method chunk is an autonomous, cohesive and coherent part of a method.
  - A method may be viewed as a collection of loosely coupled method chunks.

[Ralytė & Rolland, 2001; Mirbel & Ralytė, 2006]
Template for method chunk definition (1/2)

- **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 *interoperability problem(s)* identified in the interoperability classification framework (see interoperability classification framework) that this chunk addresses.
Template for method chunk definition (2/2)

- **Reuse situation**: A set of criteria characterising the context in which the method chunk is suitable.
- **Reuse intention**: 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**:
  - **Sub-chunks**: The list of sub-chunks if this chunk is an aggregate.
  - **Product Part**: The product part of this chunk represented in terms of a metamodel and an informal description.
  - **Guideline**: The process model of this chunk.
- **References**: List of references to the literature related to this chunk.
- **Application Example**: Application examples provided in order to help the method engineer to apply the method chunk.

Example of Method Chunk (1/4)
Example of Method Chunk (2/4)

**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 & Business.Operational.Business Alignment

**Reuse situation:**  
Application domain.Application type.Inter-organisation application  
Application domain.Impact of legacy system.Functional domain reuse  
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.

Example of Method Chunk (3/4)

**Body:**  
**Product Part:** Integrated definition of products and business processes.

**Business Model**

- Business Value
  - Price
    - has value for
    - Business Actor
      - has
      - Business Benefit
        - creates
        - interacts with
        - Employee
        - Customer
        - Supplier

**Business Process Model**

- Interface
  - Flow Object
    - control flow
    - data flow
    - Task
      - Process Start
      - End
      - Decision
      - Parallelity
      - Synchronization
    - Activity
      - Sub-process

Deliverable DTG6.3
Example of Method Chunk (4/4)

Body:
Guideline:
1. Define the product structure in accordance with the business metamodel.
2. Define the business process structure.
3. Assign the responsible business actors to the activities and sub-processes of the business process.
4. Define the interfaces which are necessary to connect the activities and sub-processes.
5. 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.

Plan of the Tutorial

- Introduction
- Situational method engineering to support interoperability

Method Chunk Repository for Interoperability
- Objectives
- Schema of the MCR
- Meta-CASE Tool
- Actors
- Services

- Interoperability classification framework
- Conclusion
Method Chunk Repository (MCR)

A tool support to collect method knowledge for interoperability in the form of method chunks

Objectives:
• To store method chunks
• To relate method chunks to the interoperability problems that they address
• To provide facilities for method chunks indexation, selection and retrieval
• To provide facilities for method chunks evaluation
• To capitalise knowledge about method chunks application in different cases

MCR Value Proposition

• 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 adoption challenge.
• Network effects complicate the common return on investment and risk analyses within the Method Chunk Repository enabled Community of Practice (MCR COP).
• Concerted regulative cycles help to articulate value, problem, and solution for all entities involved.
**Multi-level "Concerted Regulative Cycle"**

- MCR adopting entities must change their project attitudes (Regulative cycle)
- Levels or scales of these entities: individual; organisation; sector; society
- Partial Architecture Descriptions have focussed at:
  - sector/society:
    - Athena MDI Framework (DTG 6.2 Appendix 6) – collaboration platform aspects
    - chunk meta-model & problems classification
    - organizations: remedy interoperability problems
- Prototype Realizations and Cases have focussed at:
  - organisation: Insurance domain and sewage case
  - Meta Case tool

---

**Schema of the MCR**

- **Method Chunk**: situation-dependent guidelines to solve a given problem
- **Interoperability problem**: a state that is regarded to include a lack of interoperation
- **Application case**: a project that may face several interoperability problems
- **Evaluation report**: grades the merits of using a method chunk to solve an interoperability problem in a real application context

[Jeusfeld et al., 2007]
MCR as a part of a Meta-CASE Tool

**Meta-CASE Tool for Interoperability**

- Situational Method Engineering Services
- Data
- Situated Method Application Services

- Method Chunk Engineer
- Situated Method Engineer
- Classification Manager
- Method User

*MC: Method Chunk, **MCR: Method Chunk Repository

 Actors linked to the MCR

<table>
<thead>
<tr>
<th>Actor</th>
<th>Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method chunk engineer</td>
<td>To capture knowledge to specific interoperability problems as method chunks that can be used in different application cases and to characterise method chunks following the classification scheme.</td>
</tr>
<tr>
<td>Situated method engineer</td>
<td>To find a set of method chunks that can be assembled into a coherent method addressing a particular (interoperability) development/analysis need in a particular application case.</td>
</tr>
<tr>
<td>Classification manager</td>
<td>To develop and evolve interoperability classification schemes for classifying method chunks so that they are easy to search and navigate.</td>
</tr>
<tr>
<td>Method user</td>
<td>To be able easily and efficiently test/analyse/apply method chunks to specific cases, as well as describe experience of using these method chunks in his/her specific case.</td>
</tr>
</tbody>
</table>

[Jeusfeld et al., 2007]
Services linked to the MCR (1/3)

For the Method chunk engineer:

Meta-modelling facility: This facility should support the development and maintenance of meta-models that describe product and process parts of the method chunk.

Method chunk authoring: Services to support method chunk definition according to the method chunk meta-model, including:
- a graphical editor with model templates for composing method chunks,
- guidelines, templates, examples, checklists
- indexation/classification mechanism for method chunks characterisation.

Services linked to the MCR (2/3)

For the Situated method engineer:

Search facility: Services to search, select and browse method chunks in the MCR.

Method chunks assembly: Services to assemble selected method chunks into a situational method and to validate the obtained method completeness and correctness.

Case situation assessment: Services to evaluate and assess the situation of a particular industrial case including interoperability problems identification and to specify requirements for a case-specific method.
Services linked to the MCR (3/3)

For the Classification manager:
Classification management: Services to develop, administrate and evolve the classification framework.

For the Method user:
Method chunk enactment: Services to enact situational methods in specific projects including:
• Guidelines to select method chunks from the case-specific method and to apply them (instantiate product and process models).
• Graphical editor to visualise results obtained by enacting method chunks.
Method chunk evaluation: The facility to write an experience report concerning method chunk application in a particular case.

Prototype(s) of the MCR

• Earlier version of the meta model implemented in METIS
• Current version is implemented in ConceptBase

Both prototypes are however not full-fledged MCRs but they only allow to represent the content of the MCR, not the functionality.
Plan

- Introduction
- Situational method engineering to support interoperability
- Method Chunk Repository for Interoperability

**Interoperability classification framework:**
- Objectives
- Interoperability problems classification
- Case study: Insurance domain
- A meta-model for interoperability problems classification
- Applying interoperability framework
- Conclusion

---

**Interoperability classification framework**

An indexation mechanism related to the MCR

**Objectives:**

- To support method chunks characterisation – to link them to the interoperability problems that they address
- To support application case assessment – to identify the interoperability problems that occurs in this case
- To support matching between the case situation and method chunks
A laymen’s view

System A

System B

- data needs to be exchanged
- call each other’s services
- organizational borders
- socio-technical system

This is an ever changing nightmare!

An analogy

An interoperability problem is like a symptom of a patient’s disease.

The doctor has to identify the symptoms, find the root cause, and select a suitable therapy.

Every case is different but some patterns are shared and allow for building up a knowledge base.

Research question:
How to describe the symptoms (interoperability problems) and how to browse for possible therapies?
Interoperability problems classification

- A meta-model for interoperability problems classification

![Diagram of Interoperability Problems Classification]

Case: Interoperability in the Insurance Domain

- **Insurance Portal** - A common Internet platform for sales partners, e.g. agents and brokers
- **Objective**: to support independent insurance agents with a single point of access to products and services of different insurance companies
- **Advantages for the agents**:
  - single point of access to reduce cycle times for business processes such as offer, contract, and portfolio management
  - less administration costs
  - improved service quality because of a broad product and information portfolio
- **Advantages for the insurance companies**:
  - reduced maintenance and operation costs for their partner systems due to cost sharing
  - enlarged sales force because of potentially new agents
Case: Interoperability in the Insurance Domain

- New business model based on Internet technology in order to reduce administration costs and to establish new sales channels

Businesses Model

Customer -> Sales Partner -> Insurance Platform -> Sub Service Provider

Interoperability Issues in the Business Domain

- Strategic issues - definition of a business strategy of each participating partner
  - Which are the processes and services (products) to be realised on the platform: Intra- and inter-organisational?
  - Which are the appropriate business partners to develop and run the platform?
  - Does the business plan of the platform correspond with the business plans of each partner?
**Case: Interoperability in the Insurance Domain**

- **Interoperability Issues in the Business Domain**
  - Operational issues – definition of various business processes
    - Business process modelling with a special focus on the products and interfaces between the business actors involved:
      - Insurance core service processes
      - Value adding processes
      - Development processes
      - Business operations processes
    - Identification of interoperability problems and opportunities in the business domain:
      - Product Management
      - Process integration of business partners
      - Training and Learning
      - Pricing Model

- **Interoperability Issues in the ICT Domain**
  - Development issues:
    - Integration of different viewpoints of requirement definition
    - Selection and integration of implementation technologies and platforms
    - Selection and integration of different modules of the implementation environment
  - Execution issues:
    - Data conversions: customer data, contract data, product data
    - Component integration: How can different components of functionality be operated within a single business service?
    - Synchronisation and integration of long lasting transactions
Case: Interoperability in the Insurance Domain

• Summary
  • An ICT project integrating several organisations is typically characterised by a multitude of interoperability problems
  • A consistent method that will solve all possible interoperability problems does not exist because the business and ICT domains are too diverse

• Proposal
  • An extensible and domain-specific knowledge base of method chunks to support the development of interoperable systems

Ontological Dimensions for Classifying Interoperability Problems

1. From which knowledge domain can we draw expertise to understand the interoperability problem?
   ➔ Knowledge Dimension

2. During which lifecycle stage does the problem occur?
   ➔ Lifecycle Dimension

3. Which types of products are involved in the observed problem?
   ➔ Production Dimension

4. Which types of processes were active when the problem occurred?
   ➔ Process Dimension

5. Which types of human/automated producers are involved in the problem?
   ➔ Producer Dimension

[Jeusfeld et al., 2007]
Why these five questions?

- We believe that they are sufficiently general to characterize interoperability problems (people, systems, data, models, programs, phases, language issues, etc.)

- Still, we are not particularly dogmatic about the 5 dimensions. There could be less or more. But not many more as we need some uniformity to lookup entries in the MCR.

Tagging the problem

We mark the problem by tags (problem classifiers) that make the problem ‘visible’ to our counter measures.

Note that a given problem may be marked by many tags.

The tagging is a form of abstracting from problem details.

Each tag is a certain statement on the problem. Hence, all tags together tell a story about the problem.
Values for the dimensions must be taken from a standardised vocabulary [Jeusfeld et al., 2007]

Example

“Due to European legislation, the fee for sewage water has to be based on fresh water consumption. In the old system, it has been based on the number of people in the household. City administration provided citizen data in spreadsheets or text files to the sewage water company. In the new situation, the data has to be synchronized with the ERP system of the fresh water company, which has data on customers rather than on citizens.”
Answers to the five questions

1. From which knowledge domain can we draw expertise to understand the interoperability problem?
   domain.data-modeling

2. During which lifecycle stage does the problem occur?
   phase.ICT-development.system-analysis

3. Which types of products are involved in the observed problem?
   product.dataschema.customer-database, product.document.citizenlist

4. Which types of processes were active when the problem occurred?
   process.human-interaction.expert-discussion

5. Which types of human/automated producers are involved in the problem?
   producer.human.database-administrator, producer.human.accountant
Another instance

producer.human.ERP-administrator

producer.human.billing-department

producer.system.ERP

producer.system.BS

Common water bill project

chunk, remove media breaks

IP123 heterogeneous data formats

Next Steps: (1) Fill the MCR

Search for chunks:

1. Describe the situation by problem classifiers

2. Find matching interoperability problems

3. Lookup the associated method chunks
Next Steps: (2)...do it in Collaborative Platform

- Agency A (Physical World)
  - Institutional Tools
  - Institutional Transactions
  - Users
  - Institution
  - Institution rules & patterns
  - Business Transactions
  - Business Processes
  - Business Collaboration
  - Business
  - Users

- University B (Physical World)
  - Institutional Tools
  - Institutional Transactions
  - Institution
  - Institution rules & patterns
  - Business Transactions
  - Business Processes
  - Business
  - Users

Software Bus (Internet)

- Vertical Integration
- Infrastructure Services
- Registry/Repository
- Model Mgmt.
- Service Mgmt.
- Task Exec.
- Data Mgmt.
- User Interface Services
- User Services
- Institution Services
- Resource Services

Next Steps: (3) Launch a Community of Practice

- Overcoming the knowledge & technical barriers is feasible !!
- Next, the barriers to cooperative filling and using MCR must be overcome:
  - Can different knowledge institutions publish within a single MCR model and share "own" MCR results as a common pool resource?
  - Incentives matter !
  - and so do regulations (e.g. EU Database Directive)*

*http://europa.eu.int/ISPO/infosoc/legreg/docs/969ec.html
Learning by Doing/Using – Experience Matters

• Chunks with positive experience reports are ranked higher
• Over time, only the successful chunks survive
• We need an organization in which this approach is tested
• Seed: fill the MCR with generic chunks

Conclusion

• A knowledge-based approach where solutions to common interoperability problems are encoded as method chunks
• An extensible taxonomy of interoperability problems
• Application on a real-world interoperability case
• Three prototypes for the method chunk repository (two using METIS tool, one using ConceptBase)
• Adoption requires Collaboration
Future plans

- MCR concept being considered by BOC for supporting their in-house knowledge sharing
- Provide a more advanced prototype
- Fill the MCR with content!

More reading

Other References


Classifying Interoperability Problems for a Method Chunk Repository

Manfred A. Jeusfeld¹, Per Backlund², Jolita Ralyté³
¹ Tilburg University, CRISM/Infolab, 5000 LE Tilburg, The Netherlands
  Manfred.Jeusfeld@uvt.nl
² University of Skövde, P.O. Box 408, SE 541 28 Skövde, Sweden
  Per.Backlund@his.se
³ CUI, University of Geneva, CH-1211 Genève 4, Switzerland
  Jolita.Ralyte@cui.unige.ch

Keywords: interoperability problem classification, method chunk, method chunk repository

Summary. This paper proposes the structure of a so-called method chunk repository that contains instructions on how to solve interoperability problems between organizations and their information systems. We detail how interoperability problems and their solutions should be tagged in order to match them. The combination of such tagged interoperability problem classifiers forms the language to express meaningful statements about the situation in which certain method chunks are applicable to solve an observed problem.

1 Introduction

The Software Engineering Body of Knowledge (Swebok, 2004), notably a book with 200 pages mentions the word interoperability just twice, once as an example for a system requirement and the second time as the title of a standard for library data models. This stands in contrast to the challenges of globalizing economy that demands solutions for an exploding number of interoperability problems (Interop, 2006). So, is interoperability just a sort of user requirement that will emerge from the system implementation if the system developers are just careful in implementing them? We claim that this is not true because interoperability is not just achieved by a technical implementation but by addressing interoperability
problems at all stages of the interaction between multiple partners, i.e. both in the business domain and in the ICT domain.

As part of the INTEROP initiative (Interop, 2006), we aim at designing a repository that stores solution descriptions for interoperability problems. In earlier papers (Ralyté et al., 2006; Backlund et al., 2006), we have reported on how to describe solutions. We proposed the concept of method chunks originally developed for situational method engineering. In this paper, we focus on how to represent interoperability problems as exposed by application cases, i.e. situations in which interoperability problems occur.

The vast array of interoperability problems calls for a domain-dependent knowledge management approach, which takes technical as well as business and organisational matters into account. Successful solutions to interoperability problems may then be stored in the form of method chunks as proposed in Situational Method Engineering (Kumar and Welke; 1992). Project-specific methods may then be created by selecting and assembling method chunks (Ralyté and Rolland, 2001b; Mirbel and Ralyté, 2006) stored in a method repository (Brinkkemper et al, 1998; Firesmith and Henderson-Sellers, 2001; Mirbel and Ralyté, 2006). The knowledge base provided by the repository is useful when dealing with problems pertaining to the interoperability domain. In particular, we find it useful in the early stages of a project. Instead of providing one universal method our approach aims to provide a knowledge base of reusable method chunks, which can be composed to form a project specific method.

In the remainder, we first introduce the concept of a method chunk repository (MCR) and a meta-case tool for situational method engineering for interoperability (MCTI). A meta-model is developed that links method chunks to application cases via the explicit concept of interoperability problem. This meta-model represents the structure of the method chunk repository. Afterwards, we derive from example cases the classifiers for interoperability problems. A problem classifier is a kind of descriptor that relates an interoperability problem to the context in which it occurs, e.g. the life cycle phase in which it occurred. In the last section we provide guidelines for applying this classifier in characterising method chunks and identifying interoperability problems in application cases.

2 Method Chunk Repository for Interoperability

The problem of enterprise interoperability not only concerns software and technologies but also enterprise knowledge and business references that must be shared. Interoperability must be achieved on all layers of an enterprise which means that a multitude of interoperability problems and opportunities have to be resolved and designed. We claim that it is impossible to create one universal method supporting all possible interoperability issues. Moreover, we are convinced that 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. We therefore propose to adopt the ideas of Situational Method Engineering (Kumar and Welke,
which promotes the notion of reusable method component also called method fragment (Brinkkemper et al., 1998) or method chunk (Ralyté and Rolland, 2001a, Mirbel and Ralyté, 2006) and the selection and assembly of these components according to the situation of the project at hand (Brinkkemper et al., 1998; Ralyté and Rolland, 2001b).

In this work we propose situational method engineering as a means for encoding situated knowledge about achieving interoperability in the form of method chunks each of them addressing one or more specific interoperability problems. A repository-based tool has to be defined in order to support method chunks storage, indexation and retrieval. We call this tool the Method Chunk Repository (MCR). The MCR becomes useful if it is included into a collaborative meta-case tool providing services for method chunks engineering as well as for the selected method chunks enactment in a specific interoperability case. In the following sub-sections we present these three notions namely method chunk, MCR and collaborative meta-case tool for interoperability.

2.1 Method Chunk

We use the definition of a method chunk provided in (Ralyté & Rolland, 2001; Mirbel & Ralyté, 2006) and adapted to the interoperability domain in (Ralyté et al., 2006). This latest method chunk metamodel allows to link best practices for achieving interoperability to specific interoperability problems. It covers best practices from the business domain as well as from the ICT domain. The main role of a method chunk is to provide guidelines to the system engineer for realising some specific system development activity (i.e business modelling, requirements specification, design, etc.) as well as to provide definitions of concepts to be used in this activity. These two kinds of method knowledge, namely method process and product parts, are captured in the method chunk body. For example, the method chunk providing guidelines for integrating two business process models will also define the meta-model that the integrated business process model should correspond.

The descriptor part of a method chunk includes a set of attributes allowing to characterise the situation in which this method chunk is applicable. A detailed classification of these criteria related to the information systems development in general, named Reuse Frame, is proposed in (Mirbel & Ralyté, 2006). This classification framework provides criteria related to the critical information systems development aspects such as organisational (i.e. contingency factors, project management aspects, system engineering activities), human (i.e. required expertise, level of involvement) and application domain (i.e. application type, level of legacy reuse, technology). But it does not explicitly include criteria specific to the enterprise interoperability domain. In our work we extend the Reuse Frame with our interoperability problems classifier presented in section 3 of this paper. That allows us to relate explicitly each method chunk to one or several interoperability problems.

The concept of a method chunk forms a complementary approach to using patterns as proposed in (Chen, 2005). 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.

2.2 Method Chunk Repository

The prerequisite for situational method engineering is a method repository containing a large collection of method chunks. Different propositions for method repositories are given in (Saeki et al., 1993; Van Slooten and Brinkkemper, 1993; Plihon et al., 1998; Ralyté, 1999; Firesmith and Henderson-Sellers 2001; Mirbel and Ralyté, 2006). All these works focus their attention on the structure, representation and storage of method chunks but do not really consider their evaluation and their suitability in different application cases.

In our MCR, besides method chunks, we aim to capitalise knowledge related to the experience and best practices of method chunks application in specific industrial cases. Therefore, as shown in Fig. 1, the MCR stores two kinds of knowledge: the reusable method chunks and the descriptions of their application cases including experience reports and evaluation how method chunks and method chunk assemblies fit in these cases. The application cases should also be characterised by using the interoperability classification framework. The contribution of such a practical method chunks applicability evaluation is multiple. It helps:

- To improve method chunks characterisation and to specify the situation in which the method chunk applicable more precisely;
- To rank method chunks providing solution to the same or similar problems;
- To extract new method chunks from experience reports
- To identify the most applicable method chunk assemblies and to store them in the MCR as new aggregate method chunks.

Fig. 1. The method chunk repository relates method chunks to their application cases via applicability evaluation reports.

By collecting method chunks in a MCR our approach provides accessibility for method users. This is an important feature of any knowledge repository (such as a method chunk repository or a pattern repository). A flexible classification scheme,
such as we propose, addresses a number of issues concerning: tool support for creating method chunks and patterns, providing reliable techniques for access, storage, search and retrieval of knowledge as well as traceability. In particular, the evaluation reports and application cases provide information of successful application of method chunks. Hence our approach forms a complement to the pattern approach proposed in (Chen, 2005) which omits connections between patterns.

2.3 Meta-case Tool for Interoperability

Situational method construction process asks for a specific software support named Computer Aided Method Engineering Tools (CAME). According to Harmsen et al., (1994) a CAME tool should provide support for the following method engineering activities: determination and valuation of contingency factors, storage of method chunks in a method base, retrieval and assembly of method chunks, validation and verification of the obtained situational method. While there is now consensus on the functionality that a CAME tool should provide, considerable work has still to be done to achieve implementation meeting this functionality. A number of meta-CAISE products and prototypes such as Decamerone (Harmsen, 1995), MetaEdit+ (Kelly et al., 1996) and MViews (Grundy and Vanable, 1996) and Mentor (Si-said et al., 1996) have been developed which implements this functionality partially.

In this work we design a Meta-Case Tool for situational method engineering in the Interoperability domain (MCTI) including required method engineering features as well as method enactment and evaluation functionality as shown in Fig. 2 illustrating the boundary model of this tool.

As show in Fig. 2, we identify four main actors of the MCTI named method chunk engineer, situated method engineer, classification manager and method user. The first three actors use the MCTI for method engineering purpose while the last
one is an application engineer which applies the method created for a particular application case. Table 1 summarises the goals of each actor.

Table 1. MCTI actors and their goals

<table>
<thead>
<tr>
<th>Actor</th>
<th>Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method chunk</td>
<td>The goal of the method chunk engineer is to capture knowledge to specific interoperability problems as reusable method chunks that can be used in different application cases and to characterise method chunks following the classification scheme.</td>
</tr>
<tr>
<td>engineer</td>
<td></td>
</tr>
<tr>
<td>Situated method</td>
<td>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 in a particular application case.</td>
</tr>
<tr>
<td>engineer</td>
<td></td>
</tr>
<tr>
<td>Classification</td>
<td>The goal of the classification manager is to develop and evolve interoperability classification schemes for classifying method chunks so that they are easy to search and navigate.</td>
</tr>
<tr>
<td>manager</td>
<td></td>
</tr>
<tr>
<td>Method user</td>
<td>The goal of the method user is to be able easily and efficiently test/analyse/apply method chunks to specific cases, as well as describe experience of using these method chunks in his/her specific case.</td>
</tr>
</tbody>
</table>

The main use cases identified in the boundary model (Fig. 2) help us to identify services that the meta-case tool has to provide to the end-users. Besides, they also serve as a starting point for more detailed scenario descriptions of human-computer interaction and working environment of the end-users. One potential way to achieve the desired functionalities is to extend commercially available modeling tools to also cover the needs for situational method engineering. This can be achieved by creating an extension of the tool that enables the representation of methods and method chunks in terms of meta-models. Several of the major modeling tools already have some form of repository support built in and many more tools can be integrated using technologies such as e.g. Netbeans (Netbeans, 2006).

3 Classifying Interoperability Problems

The classification framework has the purpose to associate method chunks as well as application 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.

3.1 Ontological Dimensions for Classifying Interoperability Problems

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 define the dimensions 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 problem?
4. Which types of processes were active when the problem occurred?
5. Which types of human/automated producers are involved in the problem?

Knowledge dimension. Iivari et al. (2004) propose five ontological domains (Type KnowledgeDomain in Fig. 3), 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. The application domain refers to the knowledge about the application domain for which the information system is intended. 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 four phases: (1) business-strategic – the phase of a project in which strategic business decisions are made, (2) business-operational – the phase in which business activities are executed, (3) ict-development – the phase in which some ICT solution is developed, and (4) ict-execution – the time when some ICT system is performing operations. This level can be further decomposed, for example the phase ict-development.analysis is the phase in which the specification of an ICT system 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.source-code.java-program. For documents, we suggest to form specialisations 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 three kinds of processes: human-process, automated-process, and human-computer-interaction. At deeper levels,
processes like \textit{human-process.meeting.group-modeling-session} are expressed. Another example is \textit{automated-process.data-exchange}.

\textbf{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 \textit{role} characterising the responsibilities of a human actor (e.g. \textit{role.system-analyst}), \textit{team} (e.g. \textit{team.development-team}), and \textit{system} (e.g. \textit{system.tool.diagram-editor} or \textit{system.enterprise-system.crm-system}). Note that producers are observable at any lifecycle stage.

The last four dimensions are adapted from the Open Process Framework (Henderson-Sellers, 2003).

3.2 Meta-model for Interoperability Problems Classification

Fig. 3 shows the addition of the problem classifier concept to the MCR meta-model. An interoperability problem is identified and described in terms of its symptom, for example "the systems of partner 1 and partner 2 cannot exchange data". Each interoperability problem can have multiple problem classifiers linking it into the business and ICT context, i.e. the universe of terms that stakeholders use when talking about interoperations of systems. The problem classifier (Fig. 3) provides a finer-grained scheme than the one utilized in (Chen et al., 2006). Therefore we expect the problem classifier to be of use when assessing interoperability problems.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig3.png}
\caption{Meta-model for interoperability problems classification}
\end{figure}

The problem classifiers associated to interoperability problems are standardised statements about the situation in which the interoperability problem has been observed or can be observed. The allowed tags are from a controlled ontology of keywords for interoperability (see section 3.1). Each individual problem classifier
is thus a viewpoint on the problem. The combination of all classifiers associated to
the same interoperability problem characterises the problem in a comprehensive
way. The restricted vocabulary for the five dimensions supports keyword-based
search for method chunks and application cases but goes beyond it. The user can
supply keywords from the five dimensions and the MCR shall respond by those
interoperability problems whose problem classifiers match the supplied keywords.
Since a problem classifier is a statement about an observation, it can be checked
in how far it is true in the context of the user.

For example, an interoperability problem may have occurred in an application
case where a cross-organisational team negotiated a contract about a cooperation
involving linking the IT systems of the organization. Then, one problem classifier
is given by the 5 values: process type=human-process.negotiation, product
type=document-type.contract, knowledge domain=organisational-domain, lifecycle
value=business-strategic, producer type=team.interorganisational-team. The users
current situation could be that there is a problem with producing a contract
between multiple partners. The match with the list of existing problems classifiers
returns all situation in which the product type is a contract and the process type is
negotiation. The user may then decide whether or not the returned problem
classifier is true in her situation as well. The fact that multiple values are combined
to a single expression is exactly the difference to simple keyword-based
approaches where any combination of keywords may be expressed regardless
whether they make sense or not. In contrast, our problem classifiers are true
statements about interoperability problems as experienced in application cases and
as successfully solved by method chunks.

4 Applying Interoperability Problems Classification

4.1 Characterising Method Chunks

Tagging of method chunks by interoperability problems is the responsibility of the
author of the chunk, i.e. method chunk engineer. 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.

4.2 Assessing Case Situation

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 method chunk selection and assembly service of the MCTI in order to
construct a case-specific method. 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, dealing with heterogenous data structures belongs to the IS domain ‘development-process’. Here, the Swebok (2004) knowledge base can be used to characterise the field.

2. Determine the lifecycle stage: Possible values are ‘business-strategic’ (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’. For example, resolving heterogenous data structures would require analysis, modelling and implementation activities in the ‘ict-development’ stage according to method chunks.

3. Determine the involved product types (if applicable). The example will involve implementaion on a specific platform in a specific language. The method chunks associated to the current problem will contain existing solutions previously classified to suit the situation.

4. Determine the involved process types (if applicable). A method chunk may be classified on the process dimension with respect to the human process of analysing the semantics of the current data structures in order to make them possible to match.

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 situated method engineer towards the most relevant interoperability issue for the case problem to be classified. The closer he/she describes the case problem along the five categories, the easier is the classification process. Furthermore, we associate experience reports of applying the chunks, which will provide the case classifier information. It will help in assessing the suitability of the method chunk in question.

5 Conclusion

The proposed approach will enhance knowledge management by means of using a method chunk repository to store reusable method chunks. The collection and storage of method chunks is supported by MCTI services for the creation, management, execution and evaluation of method chunks. In order to make
knowledge retrievable the method chunks have to be classified. The advantages of using the problem classifier are:

- It allows for more efficient retrieval of stored knowledge. This is an important feature for user services of a knowledge repository.
- It provides the possibility to use multiple classifiers makes it possible to provide a richer characterisation of method chunks
- The problem classifier is useful irrespective of how knowledge is stored, i.e. in the form of patterns or method chunks.
- A problem classifier is a meaningful statement about a situation, i.e. it is not just a combination of keywords but an expression about a past or future observation.
- The problem classifier augments the characterisation of patterns in terms of conceptual, technical and business barriers as proposed by (Chen et al., 2006).

The strength of the proposed scheme is the incorporation of organisational as well as business and technology aspects of interoperability. It also associates interoperability to existing bodies of knowledge within the information systems and software engineering domains. The proposed meta model can directly serve as the schema for an interoperability-aware method chunk repository. Prototypes based on the schema have been developed within the InterOp task group on method engineering and are currently evaluated.

References


