A knowledge-based approach to manage information systems interoperability

Jolita Ralyté\textsuperscript{a},* , Manfred A. Jeusfeld\textsuperscript{b}, Per Backlund\textsuperscript{c}, Harald Kühn\textsuperscript{d}, Nicolas Arni-Bloch\textsuperscript{a}

\textsuperscript{a}CUI, University of Geneva, Rue de Général Dufour 24, CH-1211 Genève 4, Switzerland  
\textsuperscript{b}Tilburg University, Warandelaan 2, 5037 AB Tilburg, The Netherlands  
\textsuperscript{c}University of Skövde, P.O. Box 408, SE 541 28 Skövde, Sweden  
\textsuperscript{d}BOC Information Systems GmbH, Wipplingerstr. 1, A-1010 Vienna, Austria

Abstract

Interoperability is a key property of enterprise applications, which is hard to achieve due to the large number of interoperating components and semantic heterogeneity. The inherent complexity of interoperability problems implies that there exists no silver bullet to solve them. Rather, the knowledge about how to solve wicked interoperability problems is hidden in the application cases that expose those problems. The paper addresses the question of how to organise and use method knowledge to resolve interoperability problems. We propose the structure of a knowledge-based system that can deliver situation-specific solutions, called method chunks. Situational Method Engineering promotes modularisation and formalisation of method knowledge in the form of reusable method chunks, which can be combined to compose a situation-specific method. The method chunks are stored in a method chunk repository. In order to cater for management and retrieval, we introduce an Interoperability Classification Framework, which is used to classify and tag method chunks and to assess the project situation in which they are to be used. The classification framework incorporates technical as well as business and organisational aspects of interoperability. This is an important feature as interoperability problems typically are multifaceted spanning multiple aspects. We have applied the approach to analyse an industry case from the insurance sector to identify and classify a set of method chunks.

\textsuperscript{*}Corresponding author. Tel.: +41 2223791089; fax: +41 2223797780.

E-mail addresses: Jolita.Ralyte@cui.unige.ch (J. Ralyte), Manfred.Jeusfeld@uvt.nl (M.A. Jeusfeld), Per.Backlund@his.se (P. Backlund), Harald.Kuehn@boceu.com (H. Kühn), Nicolas.Arni-Bloch@cui.unige.ch (N. Arni-Bloch).

1. Introduction

Interoperability may be seen as “the ability for a system or a product to work with other systems or products without special effort of the part of the customer” [1]. Interoperable systems have been the goal for quite some time. However, there are some obstacles in terms of technology, organisational problems and powerful technology vendors [2,3].
According to Mak and Ramaprasad [3], the basic infrastructure seems to be in place but we have not yet achieved sufficient interoperability. One important reason for this is that the problem is not only a technological one; it also resides in the organisational and business domains of information systems.

The complexity of the problem and the fact that it occurs in many different domains makes it particularly hard to address. Furthermore, we argue that interoperability is an emerging problem [4] and hence we can only see and analyse the problems as they occur in their organisational and business contexts. This means that there can be no one solution to the problem, which can be captured in a single method. We rather propose a Situational Method Engineering approach to interoperability [5–8], which will support a systematic way of analysing the project situation in order to compose a project-specific method. The situation-adapted method is composed from a number of method chunks, i.e. pieces of knowledge that has been stored in a knowledge base from which we may capitalise. However, it is not enough to store knowledge about interoperability problems in a knowledge base, it also has to be organised and tagged to cater for retrieval. This is particularly problematic in the realm of interoperability as these problems are typically ill-structured and hard to classify. Hence, we propose a way of classifying interoperability problems and their matching solutions with respect to a set of dimensions, which are useful when analysing the situation at hand. That is, we propose a knowledge-based approach to deal with interoperability problems. The benefit of this approach is that it allows for classification of the information stored in the knowledge base as well as the application of it in specific cases. This means that the knowledge base is possible to expand as it is applied to solve new problems.

This paper addresses the question of how to organise and use method knowledge to resolve interoperability problems. A specific knowledge base, called method chunk repository (MCR), is proposed as a solution to store method knowledge in terms of method chunks addressing different interoperability issues. For method chunk indexation in the repository, we propose a classification schema called Interoperability Classification Framework. This framework is also used for analysing and assessing the project situation with respect to interoperability problems. Hence, the selection of method chunks suitable to the project at hand can be done by matching chunk classification with project situation classification. Finally, in this paper, we extend the concept of the repository into a method engineering tool, named meta-case tool for interoperability (MCTI), and define requirements for such a tool in terms of services to be provided.

The paper is organised as follows. In Section 2 we introduce and characterise the interoperability domain. Section 3 describes Situational Method Engineering in the interoperability context. Section 4 presents a classification scheme for organising an interoperability knowledge repository in the form of a method chunk repository. Section 5 shows how the classification framework can be applied to analyse a case situation and to classify method chunks. Section 6 applies the proposed framework to analyse an industry case from the insurance domain. The application case shows how method chunks can be derived from practice and organised according to the proposed classification scheme. Finally, Section 7 delivers our conclusions.

2. Characterising the interoperability domain

The nature of interoperability problems and the fact that they occur in diverse domains make them hard to deal with. There is no comprehensive unified solution to be expressed as problems span technical as well as business and knowledge issues. It is rather the case that proposed solutions are scattered and focusing on different aspects of the problem depending both on the area of application and the problem domain (e.g. technological or business domain).

The interoperability problem is well known and recurring in many domains, some examples are: database schema integration [9], interoperability between modelling techniques [10], interoperability in meta-modelling platforms [11], interoperability of ERP with other systems [12], interoperability between heterogeneous information systems [13,14] and in manufacturing [15]. Furthermore, it is possible to identify large problem domains such as the healthcare domain which suffers from difficult interoperability problems as described by Dogac et al. [16]. Guijarro [17] highlights interoperability issues within the e-government domain. The e-government initiative forms an important challenge since it is a question of making citizens as well
as officials utilising societal services, which integrate systems from different agencies.

Divergence and interoperability is also a well-known problem in the open source software (OSS) community. A study by van Wendel de Joode and Tineke [18] reveals a set of strategies for dealing with interoperability issues within OSS projects. In general, two types of strategies are used: committee standardisation and market coordination. We also observe that coding style guidelines and respected gatekeepers, i.e. a knowledgeable and trusted person, are two important means for coordination [18].

Interoperability is not only a problem concerning software and technologies. There are already various technologies to realise interoperability; some examples are TCP/IP, XML, SOAP and BPEL. Dogac et al. [16] focus on web services as one potential technical solution, which is useful in the health care domain. In this context, it is worth pointing out that an extensive body of domain knowledge is defined through standards. However, Guijarro [17] stresses the fact that there is a need for guidance beyond technical issues, addressing for example organisational issues. Hence, interoperability may be described in terms of a multi-layered model consisting of a business layer, a knowledge layer and an ICT systems layer [19,1]. In order to achieve meaningful interoperation between enterprises, interoperability must be achieved on all these layers. This includes the business environment and business processes on the business layer, the organisational roles, skills and competencies of employees and knowledge assets on the knowledge layer, and applications, data and communication components on the ICT layer. In addition, semantic descriptions can be used to get the necessary mutual understanding between enterprises that want to collaborate.

Whitman and Panetto [20] identify what they refer to as pragmatic interoperability as the willingness of all partners involved to participate in collaboration. This refers to the capacity of performing requested actions as well as the policies dictating them. Similarly, Mak and Ramaprasad [3] point out that organisations must be able to contact each other using agreed protocols, share a common language, agree on goals and tasks, and have people assigned to complete these tasks in order to achieve interoperability. That is, interoperability also concerns knowledge and business references that must be shared [19].

We also draw on the experience of systems integration [2,21] to further characterise the concept of interoperability. Wainwright and Waring [22] show that the term integration is open for interpretation, as is indeed the term interoperability. There are four domains of integration: technical, systems, strategic, and organisational. The technical domain, or technical interoperability as referred to by Guijarro [17], corresponds to the ICT layer, which is further, refined into application, data and communication interoperability [23]. Johannesson and Perjons [24] propose three types of architectures for application integration: point-to-point, message brokers, and process brokers. We complement these views by making a distinction between development and execution with respect to the ICT layer (Fig. 1). The development aspect concerns all parts of the systems development lifecycle whereas the execution aspect focuses on runtime issues. Whitman and Panetto [20] distinguish between integration, as going beyond mere interoperability including some degree of functional dependency and compatibility as something less than interoperability meaning that systems do not interfere with each other’s functioning. Vernadat [25] refers to interoperability in terms

![Interoperability Diagram](image-url)
of loose integration, which puts demands on open standards. From our point of view, an integration approach is still fairly restricted to the ICT level. The main problem still remains, that systems are interoperable at all levels and that interoperability problems cannot be isolated to a particular level.

Solving interoperability problems encompasses approaches to understand the technical, strategic and organisational behaviours from a holistic perspective [22]. That is, organisations are complex and any effort has to handle all aspects in order to achieve interoperability between systems. Interoperability is a strategic issue; hence interoperability has to incorporate strategic planning for the entire system. This encompasses issues such as work practices, power and knowledge sharing which are all affected if enterprises are to be interoperable. Interoperability between two organisations is a multifaceted problem since it concerns both technical and organisational issues, which are intertwined and complex to deal with.

Interoperability is to be facilitated by combining knowledge concerning architectures and enabling technologies (to provide implementation frameworks), enterprise modelling (to define interoperability requirements) and ontology (to identify interoperability semantics of enterprises) [1]. The three knowledge domains identified by NoE INTEROP [1] have been further analysed to identify relevant interoperability problems. From the perspective of our work, we note that data integration and business process integration were identified as recurring problems. Hence, we find this issues relevant and worth pursuing from the interoperability perspective. The issues identified also highlight the fact that there is no comprehensive unified solution for interoperability problems as they reside in all domains and have effect on all layers of interoperability.

Interoperability refers to the ability of two or more organisations to exchange and interpret all necessary information to collaborate. In order for organisations to be interoperable their strategies must cater for interoperation between business processes as well as ICT systems. The business view includes the strategic and operational aspects of the business. The ICT view includes the development and execution aspects. As interoperability problems can occur in any of these aspects, or in any combination of them, they have to be analysed with respect to all combinations. Together, the views will give a description of the symptoms of the interoperability problem at hand. We summarise our view of interoperability in Fig. 1.

To conclude, true interoperability is not yet here since enterprises and organisations running different applications built with different designs and architectures still have difficulties talking to each other [3]. Furthermore, achieving interoperability also has to do with cooperative work between people from different organisations and we note that interoperability in the organisational and strategic domains also remain to be achieved in many cases.

3. Situational Method Engineering to support interoperability

Over the last decade, Method Engineering (ME), defined by Brinkkemper [26] as “the engineering discipline to design, construct and adapt methods, including supportive tools”, has emerged as the research and application area for using methods for information and software systems development. A plethora of methods have been published ranging from generic system development processes to activity- or domain-specific models. However, in the realm of methods we detected a lack of a cohesive body of knowledge concerning interoperability issues as characterised in Section 2, i.e. traditional methods have not managed to solve the interoperability problem. We argue that this is the case due to the inherent complexity and multifacetedness of the area that traditional ME is not able to capture in one universally applicable method. Method knowledge related to the information systems interoperability domain still needs to be formalised, managed and evaluated. We claim that the diversity of interoperability situations requires for a situation-specific approach of ME, usually named Situational Method Engineering (SME) [27].

SME promotes modularisation and formalisation of method knowledge in the form of reusable method fragments/chunks [28–32] and their storage in a method repository or a method base. Following this approach, construction of a situation-specific or project-specific method consists of three main steps: (1) project situation assessment and requirements specification for a method supporting this situation, (2) method chunks selection, corresponding to these requirements, and (3) selected method chunks assembly into a brand new method.

In this sense, we propose SME as a knowledge management application to encode successful solutions
to interoperability problems in the form of method chunks. More precisely, we consider specific method chunks dealing with interoperability problem solutions such as guidelines and models for data exchange, data integration, information logistics mapping, model transformation and comparison, etc. Besides, we claim that dynamic method chunk repository has to be constructed in order to make this method knowledge active—available for use, update and refinement. Finally, we consider that such a knowledge management infrastructure can be extended to a collaborative meta-case tool supporting not only method chunks management but also situation-specific method construction and experience evaluation. In the following, we define the notion of method chunk, the repository of method chunks and the meta-case tool for SME.

3.1. Method chunk

The notion of a method chunk was first proposed in Refs. [29,30,33,34] and defined as “autonomous, cohesive and coherent part of a method providing guidelines and related concepts to support the realisation of some specific system engineering activity” [31]. This definition then was adapted to the interoperability domain in Refs. [6,7]. This latest method chunk meta-model allow us 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. In Fig. 2, we provide a meta-model of method chunk abstracted from Refs. [6,7,29–31].

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 define concepts to be used in this activity. These two kinds of method knowledge, namely process and product, compose the Body of a method chunk. 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 guideline captured in a method chunk can be more or less rich, more or less complex and represented as an informal description or expressed by using different process modelling formalisms such as UML activity diagrams, NATURE contexts [35] or MAP graphs [36]. The detailed formalisation of a guideline can be found in Refs. [29–31]. The fact that a guideline can be complex (i.e., composed of a set of sub-guidelines) means that the corresponding method chunk can be an aggregate of a collection of smaller chunks. Indeed, the level of granularity of a method chunk is determined by its guideline. If sub-guidelines of a complex guideline can also be considered as autonomous and reusable in other context, the method chunk is considered as an aggregate one where each sub-guideline is represented by a sub-chunk.

The product knowledge of a method chunk can include several product parts representing input product(s) necessary for applying method chunk and target product(s) to be obtained by applying the chunk. This knowledge is generally expressed in terms of meta-models.

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 part(s) and the intention, i.e., the goal, that the chunk helps to achieve. In fact, it defines pre and post conditions of chunk application. Each intention is expressed following the linguistic approach proposed by Prat [37] as a clause with a verb and a target (see also Refs. [29–31]). It can also have several parameters, where each parameter plays a different role with respect to the verb. For example, in the intention “Construct a business model by using e3-value ontology”, the verb “Construct” is followed by the target “a business model” and the manner to achieve this intention “by using e3-value ontology”. It is formalised as follows: “Construct_{verb} (a business model)_{target} (by using e3-value ontology)_{manner}.”

The Descriptor part of a method chunk extends the contextual knowledge defined in the Interface with a set of characteristics that help to better locate the engineering situation in which the method chunk is useful. The Reuse Intention expresses the generic objective that the method chunk helps to satisfy in the corresponding engineering activity. It is formalised in the same manner as the intention, i.e., formally stated as verb + target + parameters. For instance, Align_{verb} (partner’s business models)_{target} (by applying value modelling technique)_{manner}.

The Reuse Situation is characterised by one or several Interoperability Problem Classifiers, which include criteria related to the critical interoperability aspects. The meta-model of this classifier is presented in Section 4 of this paper. That allows us to
relate explicitly each method chunk to one or several interoperability problems.

The descriptor also contains the information that can be useful for method chunk identification and selection such as Origin (i.e. the existing method or best practice provider, including references), information about its structure (i.e. atomic or aggregate) and a narrative explanation of the method chunk role defined by the attribute Objective. It can also include Evaluation Reports in order to help the method engineer to evaluate the appropriateness of the method chunk to a given situation.

To summarise, the Body and the Interface of the method chunk constitute the method knowledge to be reused in the construction of different situation-specific methods while the Descriptor captures knowledge about the reuse situations, i.e. the situation in which the method chunk is useful. Examples of method chunks are provided in Section 6 of this paper.

The concept of a method chunk forms a complementary approach to using patterns as proposed in Ref. [38]. The proposed template to encode interoperability knowledge includes the following fields: concern, barrier, approach, problem, knowledge, example, remark and reference [38]. Method chunks allow us to encode the same knowledge with the addition of a consistent metamodel as well as our proposed classification framework. Patterns may also be stored in a method chunk repository. However, one advantage of using a ME approach is that the pieces of encoded knowledge will be related to each other as well as to the type of interoperability problems they solve, which will facilitate their use.

### 3.2. Method chunk repository for interoperability

SME provides a solid, theoretically sound basis for creating project-specific methods as well as giving the development team “ownership” of their methodological approach [39]. The method chunks, of course, need to exist prior to construction of new methods. Typically, these are collected from best practice, theory and/or abstracted from other methods and approaches. Once identified and documented as conceptual models, they are stored in a specific database called method chunk repository. Different propositions for method repositories are given in Refs. [31,32,34,40–42]. 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.

The characterisation of a method chunk and its association to some particular situation is not always evident. We claim that method chunk’s suitability to a specific situation can only be evaluated in practice. Even more, the experience obtained by applying method chunks in different industrial projects is extremely precious and should also be capitalised as a kind of memory of the

---

**Fig. 2. Meta-model of method chunk.**

---

1. **Origin**
2. **Name**
3. **Objective**
4. **Score**
5. **Example**
6. **Guideline**
7. **Body**
8. **Product Part**
9. **Not-Chunk**
10. **Interface**
11. **Situation**
12. **Intention**
13. **Evaluation Report**
14. **Reuse Situation**
15. **Reuse Intention**
16. **Problem Classifier**
17. **ID**
18. **Components**
19. **Aggregate**
20. **Atomic**
community participating in the MCR construction. Therefore, in the MCR, besides method chunks, we aim to capitalise knowledge related to the experience and best practices of their application in specific industrial cases. 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 is applicable more precisely;
- to rank method chunks providing solution to the same or similar problems;
- to extract new method chunks from experience reports because successful solutions to interoperability generally come from experts;
- to identify the most applicable method chunk assemblies and to store those in the MCR as new aggregate method chunks.

For that reason, as shown in Fig. 3, the MCR stores two kinds of knowledge: the 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. In the same way as each method chunk in the MCR is characterised by a set of interoperability problems it addresses, each application case should also be described in terms of the interoperability problems that occur in it. The Interoperability Classification Framework presented in the next section provides an indexation mechanism allowing to identify and to associate interoperability problems to the chunks as well as to the industrial cases.

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, 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 Ref. [38] which omits connections between patterns.

To summarise, the MCR allows to provide a holistic view of interoperability problems and to collect solutions to these problems.

3.3. Meta-case tool for interoperability

The MCR becomes even more functional in SME if it is used with specific software such as a Computer Aided Method Engineering (CAME) tool. According to Harmsen et al. [43], CAME tool should provide support for the following method engineering activities: determination and valuation of project 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-case products and prototypes such as Decamerone [44], MetaEdit+ [45] and MViews [46]
and Mentor [47] have been developed which implements this functionality partially.

In this work, we design a meta-case tool for SME in the interoperability domain (MCTI) including required method engineering features as well as method enactment and evaluation functionality as shown in Fig. 4 illustrating the boundary model of this tool.

As shown in Fig. 4, 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 purposes; they are experts in method engineering while the last one is an application engineer, who applies the method created for a particular application case and evaluates it. Table 1 summarises the goals of each of them.

The main use cases identified in the boundary model (Fig. 4) help us to identify services (summarised in Table 2) 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 modelling tools to also cover the needs for SME. 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 modelling tools already have some form of repository support built in and many more tools can be integrated using technologies such as e.g. Netbeans [48].

4. Interoperability problems classification

The central purpose of the knowledge base is to match interoperability problems observed in application cases with method chunks that address the interoperability problem. A simple keyword-based approach has the disadvantage that it allows arbitrary combinations and provides no guideline on how to combine the keywords in a meaningful way, i.e. in a way where the combination of the keywords result in a meaningful statement. We propose a multi-dimensional approach where keywords are aggregated from specific dimensions with a limited vocabulary to form a statement. Some keywords describe the circumstances observed in the situation, others describe the knowledge domain from which to draw expertise about the situation, and a specific keyword describes the observed effect, i.e. the interoperability problem that is emerging from the situation. We call these statements problem classifiers.

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 classifiers, 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 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.
Table 1
MCTI actors and their goals (derived from Refs. [5,7,8])

<table>
<thead>
<tr>
<th>Actor</th>
<th>Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method chunk engineer</td>
<td>The goal of the method chunk engineer is to populate the MCR with method chunks, i.e. to formalise knowledge to specific interoperability problems as method chunks that can be used in different application cases and to characterise method chunks following the classification framework. Method chunks can be extracted from existing traditional methods or defined from scratch on the basis of domain knowledge and experience.</td>
</tr>
<tr>
<td>Situated Method Engineer</td>
<td>The situated method engineer is in charge of constructing a project-specific method for each industrial case. His/her goal is to find a set of method chunks that can be assembled into a coherent method that addresses a particular (interoperability) development/analysis need in a particular application case.</td>
</tr>
<tr>
<td>Classification manager</td>
<td>The goal of the classification manager is to develop, manage and evolve interoperability classification schemes for classifying method chunks so that they are easy to search and navigate. 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.</td>
</tr>
<tr>
<td>Method user</td>
<td>The goal of the method user is to be able to easily and efficiently test, analyse and apply method chunks to specific cases, as well as to describe experience of using these method chunks in his/her specific case. The provided experience reports should include the evaluation of the applied method chunks, especially their fitness to this case.</td>
</tr>
</tbody>
</table>

Table 2 (continued)

<table>
<thead>
<tr>
<th>Service</th>
<th>Description</th>
</tr>
</thead>
</table>
| Situated Method Engineer: Case-specific method construction—method chunk selection, assembly and validation | This service should help to analyse and assess the situation of a particular industrial case including:  
  - Description of the case following case specification template;  
  - Identification of interoperability problems in this case; and  
  - Specification of requirements for a case-specific method. |
| Search facility                                                          | This facility should include services to search, select and browse method chunks in the MCR according to particular needs. |
| Method chunks assembly                                                  | This facility is necessary to assemble selected method chunks into a situational method and to validate the obtained method completeness and correctness. |
| Classification manager: Classification, construction and evolution of the Interoperability Problems Classifier | Services to develop, administrate and evolve the classification framework (Interoperability Problems Classifier). |
| Method user: Method chunk enactment                                      | The role of this service is to help the method user to apply the case-specific method, i.e. to enact method chunks. It should provide:  
  - 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                                                  | This facility should help the method user to write an evaluation report concerning method chunk application in a particular case. |
4.1. Ontological dimensions for classifying interoperability problems

Interoperability problems occur in the technical as well as the organisational domains of Information Systems Development. These domains are used as a starting point for the proposed classification scheme. Ilivari et al. [49] propose five ontological domains, which are based on a review of the state of the art in current IS research. These five knowledge 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, here named business domain, refers to the knowledge about the application (business) 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 business 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. An interoperability problem can be characterised in a similar way in order to capture its multi-dimensional nature.

Interoperability problems are occurring in a certain situation within a project concerned with the interaction of multiple organisations and their information systems, hence covering both the business/organisational domain and the ICT domain. We propose to organise interoperability knowledge along the following dimensions: lifecycle, product, process and producer. The following questions define the dimensions of the classification framework:

- During which lifecycle stage does the problem occur?
- Which types of products are involved in the observed problem?
- Which types of processes were active when the problem occurred?
- Which types of human/automated producers are involved in the problem?
- Which interoperability problem has been observed is the situation?

4.1.1. Lifecycle dimension (D1)

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 as shown in Fig. 5, for example the lifecycle value ict-develop,ment.analysis is the phase in which the specification of an ICT system is analysed.

4.1.2. Product dimension (D2)

The product dimension specifies types of products that are relevant in some observed situation or that are involved in some activity. Possible product types are: document, model, data, software and hardware. Like before, specialisations are formed as shown in Fig. 6. For example, model can be all kind of diagrams, natural language descriptions or formal specifications. Some examples are model, diagram, business-model and text, scenario. For documents, we classify them into legal and technical and suggest to form specialisations according to the structure and rational of the document, e.g. document, legal, contract, sla. For documents, we classify them into legal and technical and suggest to form specialisations according to the structure and rational of the document, e.g. document, legal, contract, sla. For documents, we classify them into legal and technical and suggest to form specialisations according to the structure and rational of the document, e.g. document, legal, contract, sla. For documents, we classify them into legal and technical and suggest to form specialisations according to the structure and rational of the document, e.g. document, legal, contract, sla. For documents, we classify them into legal and technical and suggest to form specialisations according to the structure and rational of the document, e.g. document, legal, contract, sla. For documents, we classify them into legal and technical and suggest to form specialisations according to the structure and rational of the document, e.g. document, legal, contract, sla.

4.1.3. Process dimension (D3)

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 (Fig. 7), we distinguish three kinds of processes: human, automated, and human–computer interaction. At deeper levels, processes like human, individual, modelling and human, meeting, negotiation are expressed. Another example of process type is automated, data-exchange.

4.1.4. Producer dimension (D4)

Producers are human or automated actors (see Fig. 8) 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 (e.g. human.role.business-analyst) and team representing a group of actors (e.g. human.team.development). The automated producers are classified into development tools (e.g. automated.development-tool.CASE) and enterprise applications (e.g. automated.enterprise-application.crm-system). Note that producers are observable at any lifecycle stage.

4.1.5. Interoperability problem dimension (D5)

This dimension (see Fig. 9) characterises the observed problem that is the effect of some factors explicitly or implicitly included in the other dimensions. While the problem is not necessarily a logical consequence of the situation, we assume that it is statistically related to the situation.

The lifecycle, product, process and producer dimensions are adapted from the Open Process Framework [39] and compatible with observations by Voas [50].
4.2. Meta-model for interoperability situation classification

Fig. 10 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 method chunk and each application case 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. 10) provides a finer-grained scheme than the one utilised in
Ref. [38]. Chen et al. [38] propose three basic dimensions: interoperability concerns (business, process, service and data), interoperability barriers (conceptual, technological and organisational) and interoperability approaches (integrated, unified and federated), which are complemented by a set of complementary dimensions. The interoperability engineering dimension comprises of requirements, design and implementation. The interoperability measurement dimension has three values: potentiality, compatibility and performance. Finally, there is the interoperability knowledge dimension consisting of the values conceptual and technical. We note that our proposed meta-model has a more detailed classification scheme for the lifecycle value, ranging from business-strategic to ict-execution. The dimensions process type and product type allows for fine-grained classification of what the interoperability problem concerns, i.e. which products (data, documents, models) and which processes were active when the interoperability problem was observed. The producer dimension specifies which human or system actors were involved when the interoperability problem was observed.

Problem classifiers are linked to application cases, i.e. real project situations in which some interoperability problems are observed, and to method chunks, i.e. possible solutions tackle a project situation featuring some interoperability problems.

From a knowledge representation viewpoint, problem classifiers associated to method chunks and application cases are standardised statements about the situation in which the interoperability problem has been observed or can be observed: D1, D2, D3, D4 have effect D5.

The observations in dimensions D1, D2, D3, and D4 are correlated to the observed interoperability problem (D5). The correlation is not a causal one but an empirical one. Problem classifiers are created in a certain situation of an application case. In the occasion of an observed interoperability problem (D5), one records the values for the other four dimensions D1 to D4, recorded by the quintuple

\[ pc = (d_1, d_2, d_3, d_4, d_5) \]

where \( pc \) is the identifier of the problem classifier and \( d_i \) are potentially empty subsets of the five dimensions mentioned in Fig. 10.

The allowed tags are selected from a controlled ontology of keywords for interoperability (see Section 4.1). Each individual problem classifier is thus a viewpoint on the application case (or method chunk). The combination of all classifiers associated to the same interoperability problem characterises the problem in a comprehensive, though incomplete 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 which problem classifiers match the supplied keywords. Since a problem classifier is a statement about an observation, it can be checked insofar it is true in the context of the user.
For example, an interoperability problem on the definition of a business term like “advance payment” 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 organisation. Then, a suitable problem classifier is given by the five values: 

\[ d_1 = \text{business-strategic.business-planning}, \]
\[ d_2 = \text{document.legal}, \]
\[ d_3 = \text{human.meeting.negotiation}, \]
\[ d_4 = \text{human.team.management}, \]
\[ d_5 = \text{semantic-problem.concept-conflict}. \]

The user’s 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 situations 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.

4.3. Matching application cases to method chunks

The problem classifiers have the purpose to describe both application cases and method chunks by the same type of statements, specifically aggregates of tags from the six dimensions. The common description language allows matching and querying method chunks and application cases in several ways, exploiting the specialisation structure between the values for the six hierarchies. This gives the opportunity to formulate the matching problem on the basis of the common problem classifier structure. In the sequel, we assume that application cases are referenced in the repository by identifiers \( a_1, a_2, \ldots \). Method chunks are identified by \( m_1, m_2 \) and so forth. The following definition defines the problem of matching an application case \( a \) with a method chunk \( m \):

**Matching problem:**

Let \( A = \{a_1, \ldots, a_n\} \) be set of all application cases and \( M = \{m_1, \ldots, m_k\} \) be the set of all method chunks. Then, a function 

\[ f: A \times M \rightarrow R^+ \]

is called a matching function. \( R^+ \) stands for the set of non-negative real numbers. The matching problem is to find for a given application case the method chunk \( m \) such that \( f(a, m) \) is maximal.

The interpretation of \( f(a, m) \) is the degree of similarity between the problem classifiers of application case \( a \) and method chunk \( m \). The higher the value, the greater is the match between the two. Since method chunks are intended for re-use, their problem classifiers are supposed to be more generic, either by using more general values from the dimension hierarchies of Section 4.1, or by featuring more problem classifiers than an average application case. Application cases, on the other hand, are project-specific by nature. Typically, they are entered into the repository when a specific interoperability problem has been encountered during some phase in the lifecycle of the participating systems and organisations. Consequently, we expect\(^1\) that an application case has just one or at most a few problem classifiers attached to it and that the tags of these classifiers are rather specific.

The problem classifiers introduced in the preceding chapter are associated to both method chunks and to application cases. Let \( T \) be the set of all tags for allowed in the five dimensions and \( P(T) \) be its power set. Then, the function \( pc \)

\[ pc : A \cup M \rightarrow P(T) \]

shall return all tags associated to a method chunk or application case. A very simple definition of a matching function would then be the Jaccard measure

\[ f(a, m) = \frac{|pc(a) \cap pc(m)|}{|pc(a) \cup pc(m)|}. \]

It returns the percentage of shared tags between the application case \( a \) and the method chunk \( m \). This definition is assuming that the same problem classifiers that are used to tag application cases are also used to tag method chunks. This is unrealistic since method chunks are written for re-use (hence are rather general) and application cases are typically specific. A solution to this mismatch in abstraction is to introduce a similarity measure between problem classifiers themselves and to

\(^{1}\)The expectation is solely based on the well-known trade-off between relevance and reusability. It is not a pre-condition for the applicability of our approach, however.
formulate the matching functions in terms of the similarity of the problem classifiers.

We propose the following formulas to compute this similarity:

$$\text{similarity}(p_1, p_2) = \sum_{d \in D} \sum_{v_1 \in \text{tags}(d, p_1)} \sum_{v_2 \in \text{tags}(d, p_2)} \frac{1}{\text{dist}(v_1, v_2) + 1}.$$ 

The function $\text{tags}(d, p)$ returns the tags of a problem classifier $p$ in dimension $d$. The set $D$ is the set of dimensions proposed in Section 4.1, i.e. $D = \{D_1, D_2, D_3, D_4, D_5\}$. The function $\text{dist}$ is the distance (length of shortest path) of the two tags $v_1$ and $v_2$ in the specialisation hierarchy of dimension $d$ as described in Section 4.1. Hence, the bigger the distance, the smaller the similarity between two problem classifiers will be. The minimum similarity between two problem classifiers is 0 (if one of the two classifiers has no value for any dimension). The maximum value is the number of tags of $p_1$ (resp. $p_2$) if $p_1 = p_2$. The revised matching function sums up the similarities of all problem classifiers attached to the method chunk $m$ and the application case $a$:

The minimum matching value is again 0, the maximum is reached when both $a$ and $m$ are tagged by the same problem classifiers. As an example, consider the four problem classifiers, each containing tags for the five dimensions:

- $p_{c1} = \text{(business-planning, business-model, negotiation, business-analyst, view-alignment-problem)}$;
- $p_{c3} = \text{(ict-execution, data-message, data-exchange, enterprise-application, service-unavailable)}$;

The classifier $p_{c1}$ expresses that business analysts were faced with a problem to align their business models during the negotiation process while being in the life-cycle phase business planning. The second classifier is about the same type of interoperability problem but observed during the operational business process modelling phase and involving two types of products. The third classifier is about a failure to call a service while being in the process of data exchange. The involved product is a data-message and the problem is observed during ICT execution. Assume that a method chunk is tagged by $p_{c1}$ and $p_{c2}$, and an application case is tagged by $p_{c3}$. Finally, $p_{c4}$ expresses that some view alignment problem was experienced concerning the product data model.

Applying the similarity function to $p_{c1}$ and $p_{c2}$ yields

$$\text{similarity}(p_{c1}, p_{c2}) = \frac{1}{5} + \frac{1}{4} + \frac{1}{3} + \frac{1}{5} + \frac{1}{1} = 2.98.$$ 

The other values are summarised in the table below.

<table>
<thead>
<tr>
<th>Similarity</th>
<th>$p_{c1}$</th>
<th>$p_{c2}$</th>
<th>$p_{c3}$</th>
<th>$p_{c4}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_{c1}$</td>
<td>–</td>
<td>2.98</td>
<td>0.89</td>
<td>2.08</td>
</tr>
<tr>
<td>$p_{c2}$</td>
<td>2.98</td>
<td>–</td>
<td>1.02</td>
<td>2.83</td>
</tr>
<tr>
<td>$p_{c3}$</td>
<td>0.89</td>
<td>1.02</td>
<td>–</td>
<td>0.88</td>
</tr>
<tr>
<td>$p_{c4}$</td>
<td>2.08</td>
<td>2.83</td>
<td>0.88</td>
<td>–</td>
</tr>
</tbody>
</table>

Assume we would have an application case tagged by $p_{c4}$ and a method chunk tagged by $p_{c1}$ and $p_{c2}$. Then their match would be $2.08 + 2.83 = 4.91$. A method chunk tagged by $p_{c3}$ would just score 0.86 against the application case. The matching function
is strongly dependent on the definition of the dimension hierarchies. A too flat hierarchy causes relatively high match values. Another issue is the presence of multiple tags for the same dimension, e.g. pc2 has two tags for dimension D2. The more tags a problem classifier has in a given dimension, the higher will be the similarity. Still, the matching function distinguishes strong matches from weak matches.

Sophisticated matching algorithms have been investigated among others for schema and ontology matching [51]. Compared to those approaches, our matching problem is specific and simple: rather than matching the structured representations of method chunks and applications, our approach is comparing the abstract problem classifiers that are based on a very restricted vocabulary rather than allowing arbitrary expressions.

Matching is not the only way to utilise the problem classifiers; queries are another service enable by the problem classifiers. Assume that an interoperability problem format-difference has been observed, i.e. that two systems fail to interoperate because the data sent by the one system cannot be decoded by the other system. Rather than matching the complete application case, one can query for those method chunks that use representation-problem.format-difference as value for the interoperability problem dimensions. If too many are returned, one can provide further values to the other dimensions. This is a way of browsing the method chunk repository much like the navigation in data cubes of a data warehouse.

4.4. Prototype implementation

The meta-models for method chunks and interoperability problem classifiers have been developed with the help of prototypes using two meta-modelling tools, Metis [52] and ConceptBase [53]. Metis is a meta-modelling tool mainly used in modelling enterprise architectures. It provides a rich library of pre-defined modelling constructs, such as for activities, decomposition and so forth. Our experiments showed that the meta-model for method chunks can well be represented with the capabilities of the Metis constructs.

The second prototype on basis of ConceptBase has the goal to check the correctness of the meta-model against the axioms defined in ConceptBase, e.g. for correct attribution, instantiation, and specialisation. The test was successful and actually provided input for iterations of the meta-model. In particular, the integration of the problem classifiers with the method chunk model raised the question how the “product part” and the product dimension (D2) are related. In fact, they are not formally related because the product dimension is about an ontological hierarchy of types of products (regardless their representation), and the “product parts” in method chunks are indeed subject to some representation. An excerpt of the ConceptBase specification of the meta-models is provided in the Appendix. It can directly be used to store method chunks, application cases, and problem classifiers in the ConceptBase repository. It is however not an implementation of the full meta-case tool as presented in Fig. 4. It is the storage component of the tool.

The matching function is has not yet been implemented. Given its simple nature, it would take not more than a day to have it working with the prototype. Indeed, the prototype development is not a real challenge; the real challenge is to start the social process of filling the method chunk repository with content. Subsequently, we discuss this for the two elements method chunks and application cases.

5. Applying the Interoperability Classification Framework

The role of the Interoperability Classification Framework is twofold:

1. To help the method engineer to characterise method chunks with regards to the interoperability problems that they are addressing. This consists in associating one or several problem classifiers to each method chunk to be put in the MCR.
2. To analyse and assess the situation of any industrial case/project related to the enterprise systems interoperability. This also means associating several problem classifiers to the case.

In this section, we discuss how the classification framework can be used in both of these cases. The problem classifier forms a structure through which an interoperability situation can be characterised or a method chunk can be classified according to a set of predefined values as presented in Section 4 and shown in Figs. 5–9 or by adding new values.
Each method chunk in the MCR aims to provide a solution for some specific interoperability problem. 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 generalisation of a successful solution to a case problem. Then, the interoperability problem will have been stored in the MCR as result of classifying a case. Therefore, the method chunk classification process is driven by the interoperability problem definition. As shown in Fig. 11 representing our process model\footnote{We use the MAP [36] process modelling formalism to represent the process models for method chunk classification (Fig. 11) and case assessment (Fig. 12). According to this formalism, a process model is expressed as a graph where nodes represent intentions to be achieved and arks represent different strategies to achieve these intentions.} for method chunks classification, we propose first to define the interoperability problem that the method chunk deals with. The evaluation can be done by selecting a value from the interoperability problems ontology shown in Fig. 9 or by defining a new value. Next, we propose to specify the context in which the method chunk deals with this problem, that is to define in which lifecycle phase the problem is considered and to evaluate, if necessary, other dimensions (producer, product and process). All values can be selected from the corresponding hierarchy or defined as new one. The selection and definition of the values can be refined until the most fitting value is defined.

A case/project in the context of the MCR is a situation of a user (or group of users) that includes one or more interoperability problems that require to be addressed in a structured way. The process model for case/project assessment is slightly different from the process for method chunks classification as the main purpose here is to find the interoperability problems that occur in this particular case. We consider that the case evaluation process is driven by the lifecycle value. As shown in Fig. 12, we recommend for each lifecycle phase relevant in this particular case to characterise the situation by specifying the values of the optional dimensions (product, process and producer dimensions) and to identify the interoperability problems relevant in each characterised situation.

Case assessment 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. Following our approach, the method engineer is guided by a set of questions to assess cases and tag method chunks.

- **In which stage of the system lifecycle does the problem occur?** This step results in a coarse description of what type of problem is to be solved and guides the method user towards a relevant selection of method chunks. Question a–d are used to further narrow down the scope of the problem with respect to the four possible lifecycle values.
  (a) **Is the problem encountered when strategic business decisions are to be made?** The value “business-strategic” refers to the interoperability problem encountered concerns business decisions of a strategic nature such as, for example, whether the business models are compatible.
  (b) **Does the problem concern everyday business operations?** The value “business-operational” refers to whether the business processes of the cooperators are compatible and how they can be made compatible. This may typically concern cross-organisational business processes.
  (c) **Does the problem concern the development of the technical system?** The “ict-development” value tags problems that should be addressed during development of a system. This may refer to design of new systems as well as composition of new systems from existing ones. An example may be resolving heterogenous data structures when merging databases.
  (d) **Does the problem occur during runtime?** The “ict-execution” value refers to the usage situation of the ICT system. One example is the execution of communication protocols between systems.

- **What type of products are involved in the problem situation?** (if applicable). This may involve implementation on a specific platform in a specific language. A method chunk may also be associated to business documents classified to suit the situation.
- **In which process does the problem occur?** (if applicable). A method chunk may be classified in the process dimension with respect to the human, organisational or technological process it refers to. An example of a human process is analysing the semantics of the current data structures in order to make them possible to match. An organisational process may be to place an order and an example of a technological process is a client computer communicating with a server using a certain protocol.
- **Which are the stakeholders involved in or concerned by the problem?** (if applicable). The producer type value characterises aspects of the
stakeholders, involved organisations, team composition, and tools used for production. An example of a stakeholder is the managers engaged in a negotiation.

- **Determine the interoperability problem:** The set of problems is built 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. When a certain combination of chunks has been used in a project the method user can add an evaluation of them in the particular case. In the cases that a suitable method chunk cannot be found the MCR provides services for extending the knowledge base by adding new chunks to the repository.

6. **Case study from the insurance domain**

We applied the presented concepts in an industry case in the insurance domain, where competing companies joined forces to create and operate an insurance portal to share development and operation costs. In the following, we describe this industry case based on the associated business model. We then analyse interoperability problems on the strategic business level, on the operational business level and the ICT level. Based on the analysis results, we present three method chunks to be used in SME addressing interoperability.

6.1. **Case description and 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") [54]. 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.

Fig. 13 describes the business model of this industry case, i.e. how the different business participants interact with each other to create business value.

**Customers** interact with their **sales responsibles**, 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 provides 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).

All interactions within this business model raise issues concerning interoperability. To structure these issues, we use three of the interoperability
domains proposed in chapter 2 (see Fig. 1), namely the strategic business domain, the operational business domain, and the ICT domain including development and execution aspects.

### 6.2 Interoperability problems in the strategic business domain

In the strategic business domain, the business model and the associated 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 1 day, insurance company B in 7 days).

### 6.3 Interoperability problems 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.

**6.4. Interoperability problems 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 meta-models 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 instructions 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?

**6.5. Method chunks for interoperability applied in the insurance case**

We followed the procedure presented in Fig. 11 to use the MCTI in the context of the insurance case and to apply the interoperability method chunks stored in the MCR. This procedure follows the phases of the lifecycle dimension of the interoperability problem classification framework (Section 4). First, the business strategy has been defined. Therefore, the method chunk representing the Business Value Model was used. In the next step, the inter-organisational business processes were defined. They are executed between business partners involved in the business model. For this, the Product Process Dependency Chunk was applied. Then, the interaction processes of the insurance platform were described and an architectural design of the business application was done. For this, the B2B Architecture method chunk was used. Finally, IT infrastructure was designed, in which the insurance partner platform will be deployed and executed [54] (Fig. 14).

In the insurance case, we have used various interoperability method chunks relevant on different levels. Due to the lack of space, in the following we present only three of them. We have selected one from the strategic business domain (“Business Value Model” method chunk), one from the operational business domain (“Product Process Dependency” method chunk) and one from the ICT domain (“B2B Architecture” method chunk).

**6.5.1. Strategic business method chunk: Business Value Model**

Different business partners join together to cooperate in a certain business situation. A potential problem is to have competing situations during business process execution, e.g. if the cooperation
partners are competitors such as in the insurance case. To avoid competing situations, the business models have to be aligned. Below we present the Business Value Model method chunk. The product part of the method chunk is based on the e3-value ontology from Gordijn and Akkermans [55].

**Chunk ID:** MC01  
**Name:** Business Value Model  
**Objective:** Provide common language to negotiate and align value exchange of a set of cooperating actors within a business model  
**Type:** Aggregate  
**Origin:** BOC Information Systems, Ref. [55].  
**Reuse situation:** Interoperability problem = semantic-problem.views-not-aligned.business-models-not-aligned; Lifecycle value = business-strategic.business-planning;  
Producer type = human.role.business-analyst;  
Product type = model.diagram.business-model;  
Process type = \{human.meeting.negotiation; human.individual.modelling\}.  
**Reuse intention:** Alignverb (business models)target (by applying value modelling technique)manner.  
**Interface:**  
**Situation:** Business models of cooperating business partners.  
**Intention:** Defineverb (common business value model)target (with e3-value Ontology)manner.  
**Body:**
**Product part:** A meta-model for business value modelling based on ε³-value Ontology [55] (Fig. 15).

**Guideline:**
- Identify the actors, i.e. the business partners, involved in a certain business situation. Based on the market segments, where the actors are working in, complementary or common value offerings will be identified.
- For each value offering, the value port and the associated value objects will be described. Additionally, for each value port, the value exchanges and their underlying basic value transactions will be described.

**Application example:** An application example of this method chunk is the alignment of business models of different companies. The insurance companies and the platform company have to decide, how the financial flow will be executed, when a sales partner has sold an insurance product. As the insurance companies already have financial information systems to support invoicing and settlement processes, to support these processes directly by the insurance portal would have led to a re-implementaton of the corresponding components on the portal platform. Additionally, the implementation has to consider interoperable interfaces between the insurance companies and the platform. Considering this, it was decided not to offer invoicing and settlement by the platform, but to leave it at the insurance companies.

6.5.2. Operational business 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 proposes a solution for this kind of interoperability problem.

**Chunk ID:** MC02
**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

Reuse situation:
Lifecycle value = business-operational.business-modelling;
Producer type = human.role.business-analyst;
Product type = {model.diagram.product-model, model.diagram.business-process-model};
Process type = human.individual.modelling.

Reuse intention: Align\_verb (business product and business process definitions)\target (by using integrated business product and process modelling language)\manner.

Interface:
Situation: Business products and business processes of partner enterprises.
Intention: Construct\_verb (integrated business product and process model)\target (following BOC integrated business modelling approach)\manner.

Body:
Product part: Integrated definition of products and business processes [54] (Fig. 16).

Guideline:
- Define the product structure in accordance with the business meta-model.
- 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.

6.5.3. ICT method chunk: B2B Architecture

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

![Fig. 16. Meta-model for integrated definition of products and business processes.](image-url)
platform. The method chunk described below shows a general architecture to be used in such as context.

Chunk ID: MC03
Name: B2B Architecture
Objective: To provide a general architecture for a collaborative Internet-based partner platform
Type: Aggregate
Origin: BOC Information Systems
Reuse situation:
- Interoperability problem = semantic-problem.views-not-aligned.architecture-model-not-aligned;
- Lifecycle value = ict-development.design.architecture-design;
- Producer type = human.role.designer;
- Product type = model.diagram.architecture-model;
- Process type = human.individual.modelling
Reuse intention: Designverb (architecture of a common Internet-based platform)target (with service oriented approach)manner.

Interface:
- Situation: Information systems of the involved companies, the interfaces and inter-relationships.
- Intention: Defineverb (building blocks for a B2B system)target (by applying BOC service-driven model).

Body:

Product part: General software architecture of a B2B platform (Fig. 17). The arrows depict the different places of interoperability.

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,

![Fig. 17. General software architecture of a B2B platform.](image-url)
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.

As mentioned in Section 5, the method chunk engineer is allowed to extend the dimension hierarchies downwards. For example, the tag `semantic-problem.views-not-aligned.architecture-model-not-aligned` is a below the tag `views-not-aligned` in Fig. 9. The method chunks reported above are derived from a real application case, i.e. they are solving at least the case for which they were developed. Such method chunks are associated to further context information about the application case, summarised here in the method chunk body.

7. Conclusion

The increasing number of projects related to the interoperability of enterprise information systems and the critical value of these projects require specific methodological support in order to better manage their complexity and risk. However, the multitude of interoperability situations (problems, opportunities, specific requirements and other issues) cannot be addressed by one universal method to interoperability. In this work, we claim that a situation-specific method have to be created for each interoperability case in order to better fit its situation.

The approach that we propose in this paper provides a way to create and organise method knowledge to resolve interoperability problems. The method chunk meta-model serves to formalise method knowledge in terms of method chunks and supports their assembly into different situation-specific methods. Method chunks are stored in a method chunk repository, i.e. a kind of knowledge base, which makes it possible to capitalise from experience from past interoperability projects by reusing method chunks in different cases.

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. For this purpose, we define an Interoperability Classification Framework based on the notion of interoperability problem classifier and its six dimensions: knowledge domain, lifecycle value, product type, process type, producer type and interoperability problem. A simple process model guides the application of the framework for classifying method chunks as well as for assessing case situation by associating them to corresponding problem classifiers. The advantages of the classification framework are:

- It allows for more efficient retrieval of stored knowledge in the repository. This is an important feature for user services of a knowledge repository.
- It provides the possibility to use multiple classifiers and makes it possible to provide a richer characterisation of method chunks.
- It is useful irrespective of how knowledge is stored, i.e. in the form of patterns or method chunks.
- It provides a means to analyse and assess case situations. A problem classifier is a meaningful statement about a situation, i.e. it is not just a combination of keywords but an expression about past or future observations.
- It makes it possible to match interoperability problems observed in application cases with methods chunks providing solutions to these problems. That enables a better knowledge representation by storing not only solution but also information on problems.

The strength of the proposed framework is the incorporation of organisational as well as business and technology aspects of interoperability. This is illustrated by the industrial case study in the insurance domain. The framework 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. Furthermore, it can easily be extended to include other interoperability aspects. Prototypes based on the schema have been developed within the INTEROP task group on method engineering and are currently evaluated.

The project has made a partial implementation of the proposed MCR in order to help check the
correctness of the meta-models. The implementation validates the meta-model and it has been shown that an MCR can be built on top of an existing knowledge representation tool. This is a feasible approach as the implementation of repository services will be enhanced. We have tested the tool support for meta-modelling (Table 2) and method chunk authoring. We also have limited support for method chunk assembly. The classification scheme proposed in this paper has not been implemented yet and hence not the services used by the classification manager. In all, the experiments show the feasibility of our approach as well as the potential of a full implementation. In particular, the possibility to implement an MCR on top of an existing knowledge representation tool proves the feasibility of a method engineering approach to organise and use knowledge for resolving interoperability problems.

Future work will be dedicated to the specification and development of the MCTI services in order to design and develop a complete tool support for Situational Method Engineering for interoperability. Furthermore, the software company BOC Information Systems GmbH is adopting our approach for their in-house knowledge sharing of method chunks. A partial implementation based on Lotus Notes is already in daily use. The example method chunks were actually taken from this source.

Acknowledgements

This research has been carried out within the INTEROP Network of Excellence. Commission of the European Communities under the sixth framework programme (INTEROP Network of Excellence, Contract No. 508011, http://www.interop-noe.org). This work was partially supported by the Swiss National Science Foundation; Grant No. 200020-111748. The authors also wish to express their gratitude to INTEROP Task Group 6, in particular, Brian Elvesæter, Jan Goossenaerts, Frank Lillehagen, and Jaana Wäyrynen.

Appendix. ConceptBase-based prototypical implementation of the method chunk repository

This appendix lists the definitions of the MCR using the ConceptBase repository (www.conceptbase.cc) as implementation platform. The definitions are split into three parts. The first defines simple notations for meta-modelling (essentially a subset of UML class diagrams). The second represents the core MCR meta-model as presented in the paper. Finally, the third part extends the MCR meta-model by capabilities to classify interoperability problems. This is implementing the classification meta-model.

The exercise revealed that there the core MCR meta-model and the classification meta-model are not just linked by the common concept InteroperabilityProblem but also by the concepts ProductModel/ProductType and ProcessModel/ProcessType. Hence, the exercise with implementing the structure of the MCR with ConceptBase was very useful to make the overall meta-model more cohesive.

```/*
* File: MCR-Notation.sml
* Author: Manfred Jeusfeld
* Date: 8-Mar-2006 (9-Nov-2006/M.Jeusfeld)
MCR_Concept with
  attribute
    property: Domain;
    association: MCR_Concept;
    hasPart: MCR_Concept;
    contains: MCR_Concept
end
MCR_Concept!property isA MCR_Concept end
MCR_Concept!association isA MCR_Concept end
Domain isA MCR_Concept end
String in Domain end
Real in Domain end```
MCR_Concept MCR_Element with
    property
      id: String;
      descriptor: String
    end

MCR_Concept MethodChunk isA Guideline with
   property
     score: Real
     association {* any example for exactly one method chunk *}
     hasExample: MethodApplicationExample;
     hasReuseSituation: ReuseSituation;
     hasReuseIntention: ReuseIntention;
     hasOrigin: Organisation
   end

MCR_Concept MethodApplicationExample isA MCR_Element end
MCR_Concept ReuseIntention isA MCR_Element end
MCR_Concept AggregateMethodChunk isA MethodChunk with
   hasPart
     partChunk: MethodChunk
   end
MCR_Concept AtomicMethodChunk isA MethodChunk end
MCR_Concept Method isA MethodChunk with
   contains
     containsChunk: MethodChunk;
     containsProcessModel: ProcessModel;
     containsProductModel: ProductModel
   end

MCR_Concept Guideline isA MCR_Element with
   association
     representedBy: MethodChunk;
     hasInterface: Interface;
     isBasedOn: ProductPart
   end

MCR_Concept NotChunkGuideline isA Guideline end
MCR_Concept ReuseSituation isA MCR_Element with
   hasPart
     hasCriterion: Criterion
   end
MCR_Concept Criterion isA MCR_Element end
MCR_Concept ExperienceReport isA MCR_Element with
   association
MCR_Concept Organisation isA MCR_Ele...
References


