Method Chunks for Interoperability

Jolita Ralyté[1], Per Backlund[2], Harald Kühn[3], Manfred A. Jeusfeld[4]

[1] CUI, University of Geneva, Rue de Général Dufour, 24, CH-1211 Genève 4, Switzerland
Jolita.Ralyte@cui.unige.ch
[2] University of Skövde, P.O. Box 408, SE 541 28 Skövde, Sweden
Per.Backlund@his.se
[3] BOC Information Systems GmbH, Rabensteig 2, A-1010 Vienna, Austria
Harald.Kuehn@boc-eu.com
[4] Tilburg University, CRISM/Infolab, 5000 LE Tilburg, The Netherlands
Manfred.Jeusfeld@uvt.nl

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. Platform-based approaches such as service-oriented architectures address the technical integration of systems. However, a deep integration needs to cover the whole lifecycle of the interoperable system. We propose method engineering as a means for encoding situated knowledge about achieving interoperability in the form of method chunks. We analysed the field of interoperability for enterprise applications and propose that a tool modelling the business- and ICT-related choices in the form of method chunks is needed for a knowledge-based solution of interoperability problems. An industrial case is included to back our claims.

1 Introduction

The competitiveness and efficiency of an enterprise largely depends on its ability to interact with other enterprises and organisations. Not only large organisations set up cooperation agreements with other enterprises, also small and medium sized enterprises are combining their forces to compete jointly in the market. This evolution makes interoperability between enterprises and software systems an increasingly important issue. Interoperability is one of the key challenges for modern enterprises.

The problem of interoperability is as old as the existence of software systems. A first idea was to make enterprise applications interoperable via central databases. This approach failed in practice because not enough semantics could be covered in the database schema to understand the semantics of data. As a consequence, non-interoperable applications were created based on decentralised data management. The next attempt was to save the original vision of data independence by so-called federated databases. For the same reason as for the central databases, this approach has not passed the test in practice: it is almost impossible to create a global understanding of data without referring to application semantics, let alone business semantics.

Another school of interoperability has been concerned with standardising system interfaces in a way that one system can call the other system. This has led to plat-

forms such as RPC, CORBA, J2EE and .NET, to name a few. Here, the problem of interoperability is only addressed at the technical level and fundamentally relies on information hiding. It may be perfectly feasible to call a remote service with the parameter values that are completely non-sensical.

We claim that we need a domain-dependent approach to interoperability. Rather than focusing on technical interoperability alone (which is mainly solved by industry standardisation), we propose to encode successful solutions to interoperability problems as suggested in Situational Method Engineering [13]. Some solutions deal with technical interoperability problems; others are about aligning business processes. Situational Method Engineering promotes project-specific method construction by selecting and assembling method fragments [3] or chunks [16, 20] stored in a method repository [3, 6, 16, 19] hence addressing the method requirements of the specific project. The repository then becomes the common knowledge base which can aid in interoperability solution projects. Hence, our approach can contribute in the early stages of such projects by setting up a project specific method. In this sense we envisage method engineering as a knowledge management application for projects within the interoperability domain. Instead of providing one universal method for interoperability problems solution we propose to define a knowledge base of reusable method chunks each of them addressing one or more specific interoperability problems. The latter are grouped in an extensible hierarchy of interoperability problem classes.

The remainder of this paper is organised as follows. In section 2 we characterise the field of interoperability between enterprises and systems. Section 3 presents an industrial case and identifies some associated interoperability problems. In section 4 we discuss how situational method engineering and the notion of reusable method chunks can be adopted to structure specific solutions to interoperability problems. The paper ends with a review of this work, and outlines future research.

2 Characterising the Interoperability Domain

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” [10]. 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 [15, 7]. The basic infrastructure seems to be in place [15] but we have not yet achieved sufficient interoperability. The problem is well known and recurring in many domains, some examples are: database schema integration [18], interoperability between modelling techniques [5], interoperability in metamodelling platforms [14], interoperability of ERP with other systems [1], and CNC manufacturing [23].

Interoperability is not only a problem concerning software and technologies. It is also a problem that concerns knowledge and business references that must be shared in order to achieve interoperability [4]. Hence, interoperability is described in terms of a three-layered model consisting of a business layer, a knowledge layer and an ICT systems layer. In order to achieve meaningful interoperation between enterprises, interoperability must be achieved on all layers of an enterprise. 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.

Similarly, Mak and Ramprasad [15] 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. Moreover, we may not assume that interoperability concerns only the interoperability between enterprises.

We also draw on the experience of systems integration [8, 7] to further characterise the concept of interoperability. Wainwright and Waring [24] 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 corresponds to the ICT layer, which is further, refined into application, data and communication interoperability [22]. Johannesson and Perjons [12] 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. The development aspect concerns all parts of the systems development life cycle whereas the execution aspect focuses on runtime issues.

The business and knowledge layers are further refined in the systems, organisation and strategic domains [24]. The systems domain encompasses approaches to understand the technical, strategic and organisational behaviours from a holistic perspective. 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. Finally, the organisational domain 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. We summarise our view of interoperability in Fig. 1.

There are already various technologies to realise interoperability; some examples are TCP/IP, XML, SOAP and BPEL. However, true interoperability is not yet here since enterprises running different applications built with different designs and architectures still have difficulties talking to each other [15]. Whereas achieving interoperability also has to do with cooperative work between people from different organisations. Furthermore, we also note that interoperability in the organisational and strategic domains also remain to be achieved in many cases.

Divergence and interoperability is a well-known problem in the open source software (OSS) community. A study by van Wendel de Joode and Tineke [25] 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 [25].

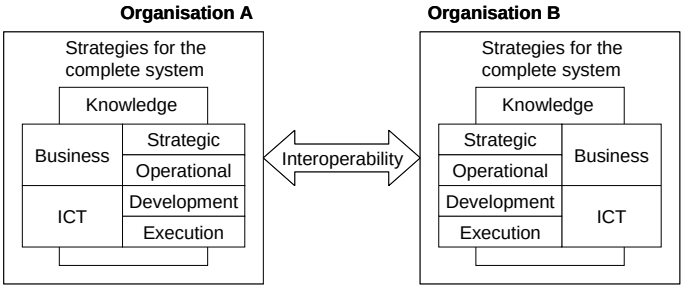


Fig. 1. Interoperability between two organisations entails interoperability in all domains

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). The three knowledge domains identified by NoE INTEROP [10] have been further analysed to identify relevant interoperability problems [17]. 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.

3 Interoperability in the Insurance Domain: a Case

In this section we analyse an industrial case of interoperability in the insurance domain and identify interoperability problems related to this case. We classify them following our characterisation framework presented in Fig. 1.

3.1 Business Model

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

The following industry case describes a *B2B sales platform for insurance partners based on Internet technology* ("insurance portal"). The main objective of the insurance portal is to support independent insurance agents with a single point of access to products and services of different insurance companies. An agent is working for several competing insurance companies on a commission basis. Some advantages for the agents are a single point of access to reduce cycle times for business processes such as offer management, contract management, and portfolio management, less administration costs, and improved service quality because of a broad product and information portfolio. Some advantages for the insurance companies are reduced maintenance

and operation costs for their partner systems due to cost sharing and an enlarged sales force because of potentially new agents.

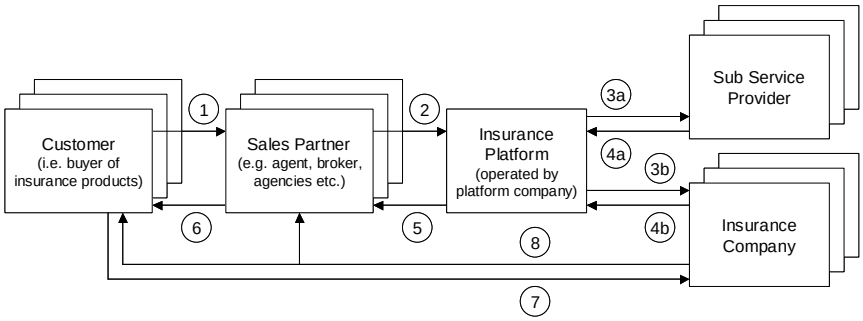


Fig. 2. Business Model from Insurance Domain based on Common Platform

Fig. 2 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 provided the best offer, and pays the insurance fee to the insurance company (step 7). The insurance company delivers the appropriate contracts, pays the commission fees, and fulfils its part of the insurance contract (step 8).

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, namely the strategic business domain, the operational business domain, and the ICT domain including development and execution aspects.

3.2 Interoperability Issues in the Strategic Business Domain

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

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

and inter-organisational business processes (e.g. application and claims processes) can be distinguished.

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

3.3 Interoperability Issues in the Operational Business Domain

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

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

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

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

- *Test Management:* In combination with the product model, a set of test cases can be developed as a specification for testing the platform application and interoperability.

3.4 Interoperability Issues in the ICT Domain

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

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

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

3.5 Summary of the Case

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

4 Situational Method Engineering to Support Interoperability

We use the term method to denote a regular and systematic way of accomplishing a result. Methods cover a wide spectrum of industrial capabilities and services incorporated in either pragmatic or scientific working methods. Moreover, we claim that a method may be decomposed into a set of method chunks [20]. In the realm of methods there is 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 multi-facetedness of the area. In this sense, we propose method engineering as a knowledge management application. Systems development has been characterised as knowledge work [11, 9]. In this context, we view the development method as a body of knowledge. In order to make it an active body of knowledge it has to be made available for use, update and refinement, something which may be achieved by constructing a dynamic method chunk repository [2, 16, 19].

In the following we will demonstrate how Situational Method Engineering can help in solving parts of the problem of managing interoperability knowledge. 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.

4.1 Method Chunk

We propose to use the notion of reusable method chunk [16, 20] to represent methodological knowledge related to interoperability. A method chunk is an autonomous, cohesive and coherent part of a method providing guidelines and related concepts to support the realisation of some specific system engineering activity. A method is viewed as a collection of loosely coupled method chunks expressed at different levels of granularity. Such a modular view of methods favours their adaptation and extension and permits to reuse chunks of a given method in the construction of new ones.

As illustrated in Fig. 3, from the engineering perspective the body of a method chunk includes two types of knowledge: the process model, also called guideline, supporting the engineer in method chunk application, and the product model defining concepts, relationships between concepts, and constraints used by the corresponding process. The structure of a guideline can be found in [16, 20]. It can be more or less rich and represented as an informal description or expressed by using different process modelling formalisms. Application *examples* can be provided in order to help the method engineer to apply the method chunk.

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

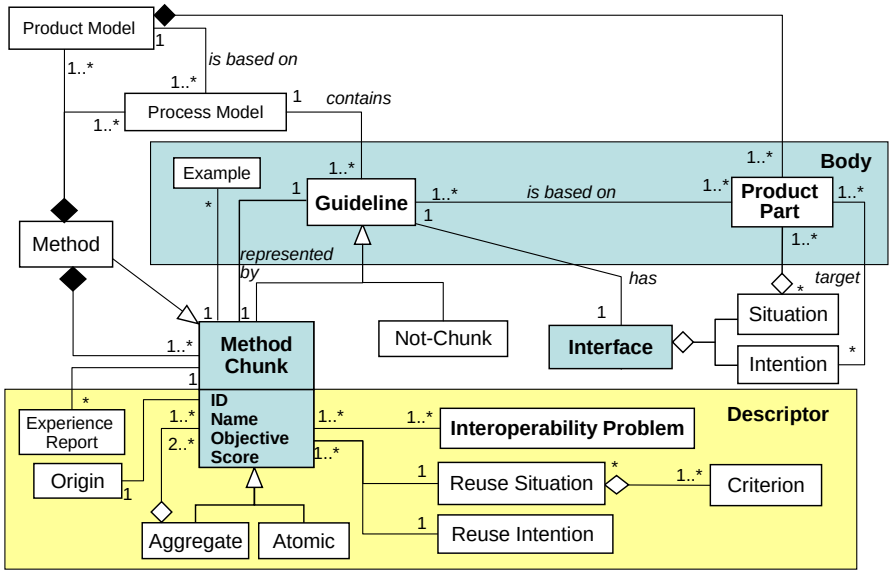


Fig. 3. Metamodel of method chunk

A set of characteristics, called a method chunk *descriptor*, is associated to each chunk in order to better situate the context in which it can be reused. The *reuse intention* expresses the generic objective that the method chunk helps to satisfy in the corresponding engineering activity. The *reuse situation* captures a set of criteria characterising the context in which the method chunk is suitable. A detailed classification of these criteria, named *Reuse Frame*, can be found in [16]. Some examples of such criteria are: system engineering activity (e.g. business modelling, requirements specification and design) in which the method chunk is relevant and characteristics of the application domain (e.g. application type, impact of legacy system and application technology). While the reuse situation and reuse intention are expressed by using keywords defined in the MCR glossary and the reuse frame, the *objective* of the method chunk provides a narrative explanation of its role.

Due to the fact that in this work we consider specific method chunks dealing with interoperability problems solution, we explicitly relate each method chunk to the corresponding *interoperability problem* identified in the interoperability classification framework illustrated in Fig. 1.

The descriptor also contains the information necessary for method chunk identification and selection such as *name, ID*, information about its structure (i.e. atomic or aggregate) and *origin* (i.e. the existing method or best practice provider). It can also include *experience reports* in order to help the method engineer to evaluate the appropriateness of the method chunk to a given situation.

4.2 Method Chunk Repository for Interoperability

In our approach, the knowledge about interoperability, based on experience and best practices or extracted from existing system engineering methods, is formalised in the form of reusable method chunks stored in a Method Chunk Repository (MCR). The process of method chunks reuse in a specific project consists of three steps: evaluating the interoperability problem at hand, selecting the appropriate method chunks from the MCR and, finally, assembling these method chunks into a situation-specific method. The last step is not tackled in this paper, see [16, 21] for details. In order to support the situation evaluation and selection process, we provide a metamodel depicted in Fig. 4 for interoperability problems definition and classification. The metamodel only shows the highest abstraction level of the classification.

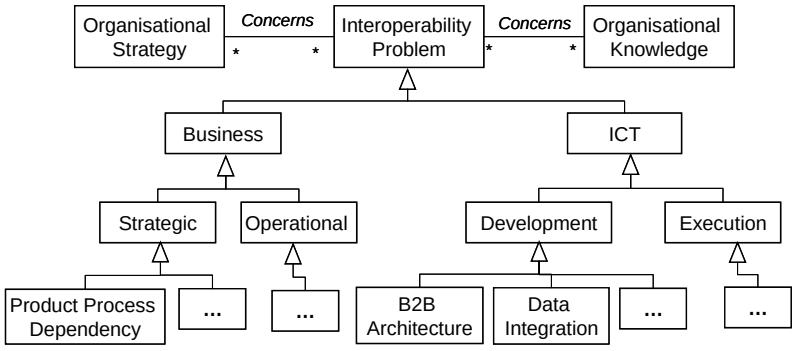


Fig. 4. Metamodel for interoperability problems classification

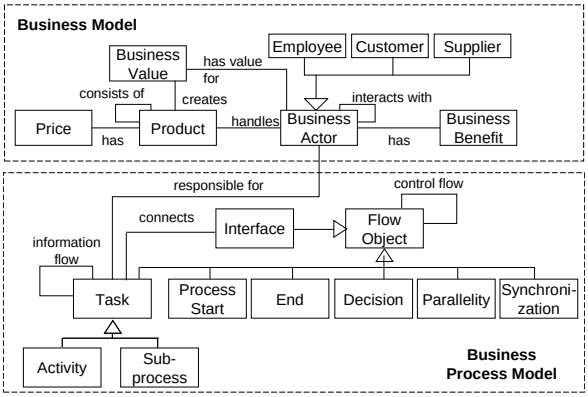
The interoperability problem identified is matched with those supported by method chunks stored in the MCR. Let us suppose the question: "How can we integrate the product data of several insurance companies?" We can identify that the interoperability problem that we are facing is classified as "ICT.Development.Data_Integration". The next step would be to ask the MCR to retrieve all method chunks associated to this interoperability problem.

4.3 Identifying Method Chunks for Interoperability: an Insurance Case

Based on the practical experiences in the insurance case we have identified several method chunks dealing with interoperability problems. Due to the lack of space, we present only two of them. Among the interoperability problems identified in section 3, we have selected one from the strategic and operational business domain and one from the ICT domain, which will be addressed to show how method chunks can be utilised to represent this knowledge.

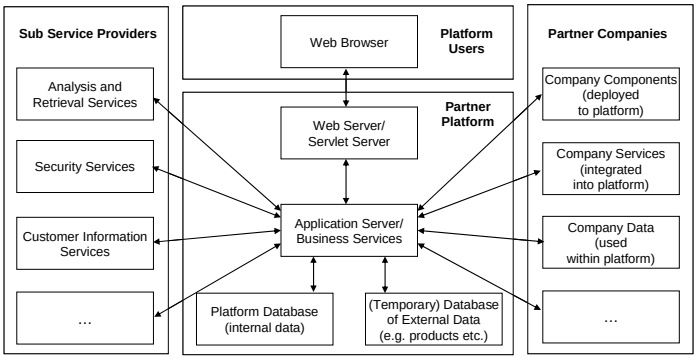
Method Chunk: Product Process Dependency

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: MC01	**Name:** Product Process Dependency
Objective: Identify dependencies between products and their corresponding business processes as basis for business alignment.	
Type: Aggregate	**Origin:** BOC Information Systems
Interoperability problem: Business.Strategic_and_Operational.Business Alignment	
Reuse situation: Application domain.Application type.Inter-organisation application Application domain.Impact of legacy system.Functional domain reuse System engineering activity.Business modelling.Business process alignment Innovation level.Business innovation **Reuse intention:** To align product definitions and business process definitions.	
Interface: **Situation:** Products and business processes of partner enterprises. **Intention:** To define integrated product and process modelling language. **Body:** **Product Part:** Integrated definition of products and business processes.  **Guideline:** Define the product structure in accordance with the business metamodel. Define the business process structure. Assign the responsible business actors to the activities and sub-processes of the business process. Define the interfaces which are necessary to connect the activities and sub-processes. By assigning the product responsibilities between products and business actors, the dependencies between products and business processes are defined transitively.	
Application Example: An application example of this method chunk is the definition of insurance products and their interdependency to business processes executed in the insurance portal. A life insurance product consists of sub-products such as risk insurance and font investment. A life insurance process consists of sub-processes such as insurance application, risk check, contracting and payment. Employees of insurance companies are responsible for executing the sub-processes. These employees are also handling several insurance products. Via this, the product process dependency is defined.	

ICT Method Chunk: B2B Architecture

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

Chunk ID: MC02	**Name:** B2B Architecture
Objective: To provide a general architecture for a collaborative Internet-based partner platform.	
Type: Atomic	**Origin:** BOC Information Systems
Interoperability problem: ICT. Development. B2B Architecture Design	
Reuse situation: Application domain.Application type.Inter-organisation application Application domain.Impact of legacy system.Functional domain reuse System engineering activity.Design Innovation level.Technology innovation; Business innovation **Reuse intention:** To establish a common Internet-based platform.	
Interface: **Situation:** The strategies, business processes and information systems of the involved companies. **Intention:** To define building blocks for a B2B system.	
Body: **Product Part:** General software architecture of a B2B platform. The arrows depict the different places of interoperability.  **Guideline:** Identify participants involved in operating and using a B2B platform. For each participant assign which of the generic building blocks are provided/used. Build an instance of each generic building block for the specific case. Describe the interrelationships within the B2B platform for each building block instance.	
Application Example: An insurance portal. The identification and assignment is as follows: *Platform users (sales agents, brokers etc.):* the sales partners access the portal via Internet and web browser technology. *Insurance partner platform:* The access of the business functionality and the generation of the user interface are via web server / servlet server. The business functionality runs on an application server. The application server stores platform internal data in the platform database. External (and temporary) data are stored in the database for external data. Via business services of the application server sub service providers and insurance companies interoperate with the insurance partner platform. *Insurance companies:* The insurance companies provide components (e.g. product calculators, risk check modules etc.), services (e.g. printing, mailing etc.), data (e.g. customer data, contract data, product data etc.), which have to interoperate with the insurance partner platform. *Sub service providers:* The sub service providers provide services such as analysis and retrieval services (e.g. data analysis, management reports, statistical evaluations etc.), security services (e.g. trust centres certificate management etc.), customer information services (e.g. credit agency services, market evaluation etc.), which have to interoperate with the insurance partner platform.	

The above two method chunks have to be seen as examples. A realistic method chunk repository shall contain hundreds of chunks of varying complexity. While the reusable chunks are formulated on generic type level, a specific case like the insurance case is formulated at a lower abstraction level, the instance level. By making the instance level explicit, the method chunk repository is extended to an experience based knowledge base (Fig. 5).

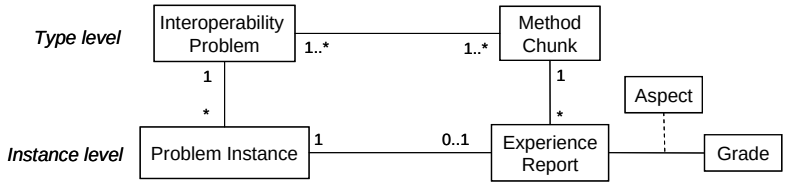


Fig. 5. Instance level in the method chunk repository

The interoperability problems from the case are formulated as a set of problem instances, which are classified into the hierarchy of interoperability problems (see also Fig. 4). When a case has been completed by executing suitable method chunks, an experience report is added to the repository that includes a critical review of the merit of the selected method chunk to solve the problem instance(s). By storing this information, subsequent cases can exploit the experience from earlier cases and select those chunks that earned high grades in the earlier cases.

5 Conclusions

Interoperability is an issue that arises when multiple organisations need to cooperate via information systems. We proposed a knowledge-based approach where solutions to common interoperability problems are encoded as method chunks. These method chunks together with experience reports and an extensible taxonomy of interoperability problems form the basis of a method chunk repository, which we are currently developing in the NoE INTEROP [10]. We see the following contributions of this paper:

- The new method chunk metamodel allows to link best practices for achieving interoperability to specific interoperability problems. It covers best practices from the business domain (e.g. aligning the business processes of enterprises) as well as from the ICT domain (e.g. integrating heterogeneous product catalogues).
- The proposed solution provides a possibility to go from generic knowledge of interoperability, via experiences of applying that knowledge, to a specific body of interoperability knowledge.
- The usefulness of the new method chunk data structure has been demonstrated by applying them to a real-world interoperability case.

A prototype for the method chunk repository is under development using the METIS tool [26] in cooperation with the METIS developers. The metamodel of Fig. 3 has been mapped to METIS meta classes. The two cases sketched in this paper have also been represented. It turned out the METIS knowledge base already had many of the business process modeling features used in the first method chunk example. For METIS, methods chunks are regarded as a substantial extension to the tool's capabilities since they are encoding procedural knowledge. The tight integration to the enterprise modelling views in METIS shall make it possible to automate parts of the method chunk execution, in particular model transformation. For non-automated parts, the system can provide assistance through the guidelines encoded in the method chunks. Additional prototypes based on the Adonis tool [27] and ConceptBase [28] are under investigation to test the general implementability of the method chunk metamodel for interoperability.

The repository will first be filled with chunks extracted by academic partners as well as the IT consulting companies to form a critical mass. Then, successful and unsuccessful applications will be added to the repository as examples of method chunks application. These examples form the experience layer of the repository. The larger the number of successful examples for a method chunk, the higher its score will be.

Future work is concerned with formalising the textual guidelines of a method chunk into a computer-interpretable process model, which allows teams from multiple collaborating enterprises to jointly execute method chunks. A related aspect is to represent the product side of method chunks, i.e. business as well as ICT models, within the repository to allow not only collaborative method chunk execution but also model sharing.

References

1. Botta-Genoulaz V., Millet P.-A. and Grabot B. (2005) A survey on the recent research literature on ERP systems. *Computers in Industry (56),* pp. 510-522.
2. Brinkkemper S. (2000) Method Engineering with Web-enabled Methods. *Information Systems Engineering: State of the Art and Research Themes.* Eds. S. Brinkkemper, E. Lindencrona, A. Sölvberg, Springer-Verlag, pp. 124-133.
3. Brinkkemper S., Saeki, M. and Harmsen, F. (1998). Assembly Techniques for Method Engineering. *10th Conference on Advanced Information Systems Engineering, CAiSE'98*. Springer, LNCS 1413, pp.381-400.
4. Chen D. and Doumeingts G. (2003) European initiatives to develop interoperability of enterprise applications — basic concepts, framework and roadmap. *Annual Reviews in Control* (27) pp. 153–162.
5. Domínguez E. and Zapata M.A. (2000) Mappings and Interoperability: A Meta-modelling Approach. *ADVIS 2000,* Ed. T. Yakhno. LNCS 1909, Springer-Verlag, pp. 352-362.
6. Firesmith D. and Henderson-Sellers B. (2001) *The OPEN Process Framework. An Introduction.* Addison-Wesley.
7. Garlan D., Allen R. and Ockerbloom J. (1995). Architectural mismatch or why it's hard to build systems out of existing parts, *Proceedings of the 17th international conference on Software engineering*, ACM Press, pp. 179-185.

8. Hasselbring W. (2000). Information system integration. *Communications of the ACM* 43 (6) pp. 32-38.
9. Hirschheim R. and Klein H. (2003) Crisis in the IS Field? A Critical Reflection on the State of the Discipline. *Journal of the Association for Information Systems,* 4, pp. 237-293.
10. INTEROP (2005) Interop Network of Excellence IST – 508011 Presentation of the Project. http://interop-noe.org/INTEROP/presentation Last accessed 2005-11-02
11. Iivari J. (2000) Information Systems Development as Knowledge Work: The body of systems development process knowledge. *In Information Modelling and Knowledge Bases X*I (Eds, Kawaguchi, E., Hamid, I. A., Jaakkola, H. and Kangassalo, H.) IOS Press, pp. 41-56.
12. Johannesson P. and Perjons E. (2000) Design principles for application integration. *Proceedings of the 12th International Conference of Advanced Information Systems Engineering, CAiSE 2000,* LNCS 1789. Eds. B. Wangler & L. Bergman. Springer.
13. Kumar, K. and Welke, R.J. (1992). Method Engineering, A Proposal for Situation-specific Methodology Construction. In *Systems Analysis and Design: A Research Agenda,* Cotterman and Senn (eds), Wiley, pp.257-268.
14. Kühn H. and Murzek, M. (2005) Interoperability in Metamodelling Platforms. In: Konstantas, D.; Bourrières, J.-P.; Léonard, M.; Boudjlida, N. (Eds.): *Interoperability of Enterprise Software and Applications*. Springer-Verlag, pp. 215-226.
15. Mak K-T. and Ramaprasad A. (2001) An Interpretation of the Changing IS/IT-Standard Game*,* Circa 2001. *Knowledge, Technology & Policy* (14) pp. 20-30.
16. Mirbel I. and Ralyté J. (2006) Situational Method Engineering: Combining Assembly-Based and Roadmap-Driven Approaches, *Requirements Engineering,* 11(1), pp. 58–78.
17. Ottoson A. (2005) *An Analysis of a Content of a Method Chunk Repository concerning Interoperability Problems*. Master Thesis HS-EA-DVA-2005-001, University of Skövde.
18. Rahm E. and Bernstein P. A. (2001) A survey of approaches to automatic schema matching. *The VLDB Journal*, 10, pp. 334-350.
19. Ralyté J. (1999) Reusing Scenario Based Approaches in Requirement Engineering Methods: CREWS Method Base. *10th Int. Workshop on Database and Expert Systems Applications (DEXA'99)*, IEEE Computer Society, p. 305-309.
20. Ralyté J. and Rolland C. (2001). An Approach for Method Reengineering. *Proceedings of the 20th International Conference on Conceptual Modeling (ER2001),* LNCS 2224, Springer-Verlag, pp.471-484.
21. Ralyté J. and Rolland C. (2001). An Assembly Process Model for Method Engineering. *Proceedings of the 13th Conference on Advanced Information Systems Engineering (CAISE'01),* LNCS 2068, Springer-Verlag, pp. 267-283.
22. Schulz K., et al. (2003) *A Gap Analysis; Required Activities in Research, Technology and Standardisation to close the RTS Gap; Roadmaps and Recommendations on RTS activities*. Deliverables D 3.4, D 3.5, D 3.6. IDEAS Thematic Network - No.: IST-2001-37368.
23. Xu X.W. and Newman S.T. (2006) Making CNC machine tools more open, interoperable and intelligent—a review of the technologies. *Computers in Industry.* 57 (2), pp.141-152.
24. Wainwright D. and Waring T. (2004) Three domains for implementing integrated information systems: redressing the balance between technology, strategic and organisational analysis. *International Journal of Information Management,* 24 (2004) pp. 329–346.
25. van Wendel de Joode R. and Tineke E.M. (2004) Handling variety: the tension between adaptability and interoperability of open source software*.* *Computer Standards and Interfaces* (28), pp. 109-121.
26. Troux Technologies (2006) http://www.troux.com/products/metis/, Metis by Troux. Online. March 30, 2006.
27. BOC Information Technologies Consulting (2006) http://www.boc-eu.com/, Adonis by BOC Online. March 30, 2006.
28. ConceptBase Team (2006) http://conceptbase.cc, ConceptBase Online. March 30, 2006.