

LABORATOIRE



INFORMATIQUE, SIGNAUX ET SYSTÈMES
DE SOPHIA ANTIPOLIS
UMR 6070

SITUATIONAL METHOD ENGINEERING: COMBINING ASSEMBLY-BASED AND ROADMAP-DRIVEN APPROACHES

Isabelle Mirbel, Jolita Ralyté

Projet EXECO

Rapport de recherche
ISRN I3S/RR-2004-25-FR

Octobre 2004

RÉSUMÉ :

MOTS CLÉS :

Ingénierie des méthodes situationnelles, composants de méthodes, itinéraires, construction de méthode par assemblage, configuration de méthode, cadre de réutilisation

ABSTRACT:

Because the engineering situation of each Information System Development (ISD) project is different, engineering methods need to be adapted, transformed or enhanced to satisfy the project situation. Contributions, in the field of Situational Method Engineering (SME), aim at providing techniques and tools allowing to construct project-specific methods instead of looking for universally applicable ones. In addition to the engineering method tailoring, necessary to fit the project situation, a customization of the engineering method for each engineer participating in the project is also required. Such a configuration allows a better understanding of the method by focusing on guidelines related to the project engineer daily tasks. It also increases his/her involvement in the engineering method realization. To achieve this twofold objective (engineering method tailoring and customization), we propose a framework for SME combining an assembly-based approach for project specific method construction and a roadmap-driven approach for engineer-specific method configuration. The first step of our process provides support to build a new method the most suitable for the current ISD project situation, whereas the second step aims at choosing the most adapted path (roadmap) to satisfy the requirements of a particular project engineer within the project-specific method. The two core elements of our situational method engineering framework are the method chunks repository and the reuse frame. The former concerns reusable method components definition and storage whereas the latter deals with the characterization of the project situation and the project engineer profile. In this paper we start first by presenting our situational method engineering framework and its core elements: the method chunk repository and the reuse frame. Then we show how to take advantage of them through our two-step process combining assembly-based method construction and roadmap-driven method configuration.

KEY WORDS :

Situational Method Engineering, Method chunk, Road-map, Assembly-based method construction, method configuration, reuse frame

Situational Method Engineering: Combining Assembly-based and Roadmap-driven Approaches

Isabelle Mirbel

Laboratoire I3S
Les Algorithmes
Route des Lucioles, BP 121
06903 Sophia Antipolis Cedex, FRANCE

Phone: +33 4 92 94 27 60

Fax: +33 4 92 94 28 96

isabelle.mirbel@unice.fr

Jolita Ralyté

Centre Universitaire d'Informatique
Université de Genève
24 rue du Général Dufour
CH-1211 Genève 4, SWITZERLAND

Phone: +41 22 379 10 89

Fax: +41 22 379 77 80

ralyte@cui.unige.ch

Abstract. Because the engineering situation of each Information System Development (ISD) project is different, engineering methods need to be adapted, transformed or enhanced to satisfy the project situation. Contributions, in the field of Situational Method Engineering (SME), aim at providing techniques and tools allowing to construct project-specific methods instead of looking for universally applicable ones. In addition to the engineering method tailoring, necessary to fit the project situation, a customization of the engineering method for each engineer participating in the project is also required. Such a configuration allows a better understanding of the method by focusing on guidelines related to the project engineer daily tasks. It also increases his/her involvement in the engineering method realization. To achieve this twofold objective (engineering method tailoring and customization), we propose a framework for SME combining an assembly-based approach for project specific method construction and a roadmap-driven approach for engineer-specific method configuration. The first step of our process provides support to build a new method the most suitable for the current ISD project situation, whereas the second step aims at choosing the most adapted path (roadmap) to satisfy the requirements of a particular project engineer within the project-specific method. The two core elements of our situational method engineering framework are the method chunks repository and the reuse frame. The former concerns reusable method components definition and storage whereas the latter deals with the characterization of the project situation and the project engineer profile. In this paper we start first by presenting our situational method engineering framework and its core elements: the method chunk repository and the reuse frame. Then we show how to take advantage of them through our two-step process combining assembly-based method construction and roadmap-driven method configuration.

Keywords: Situational Method Engineering, method chunk, road-map, assembly-based method construction, method configuration, reuse frame

1 Introduction

Objectives, problems and requirements as well as the whole engineering environment vary from one Information Systems Development (ISD) project to another. Besides, each project is governed by a number of organisational, technical and human factors as complexity, specificity and novelty of the application domain, degree of innovation, way of working, team expertise etc. In other words, the engineering situation of each ISD project is different and requires for project-specific methods and tools supporting it. Traditional Information Systems (IS) engineering methods cannot be applied as they are; they need to be adapted, transformed, or enhanced to satisfy the project situation at hand. However, they are rather rigid and there is a great difficulty to change them.

To resolve this problem, the discipline of Situational Method Engineering (SME) focuses on the creation of new techniques and tools allowing to construct project-specific methods ‘on the fly’ [Kumar92] instead of looking for universally applicable ones.

Most of the SME approaches promote the construction and adaptation of new methods by assembling reusable *method fragments* [Harmsen97, Brinkkemper98] or *method chunks* [Ralyté01a],

stored in some method repository [Saeki93; Harmsen94; Harmsen95; Brinkkemper98; Plihon98; Ralyté99b]. These approaches lead to the construction of new modular methods that can be modified and improved to meet the requirements of a given situation [Harmsen94], [Slooten93].

In addition to the tailoring of project-specific ISD methods, there is also a need for the customisation dedicated to ISD project crew members [Mirbel02, Mirbel04a, Mirbel04b]. Indeed, each ISD crew member has to perform specific tasks throughout the project at hand and thus needs specific help tailored to satisfy his/her requirements.

Therefore, we consider that each ISD project should start with the definition of its proper method that best fits its situation. Then, the obtained method should be personalized for each project crew member according to his/her tasks; only the method chunks supporting these tasks should be proposed to him/her.

The objective of our work is to propose an approach for SME including both project-specific ISD method construction and its customisation for each engineer participating in the project. To achieve this objective, we combine two complementary approaches (1) the *assembly-based* approach for project-specific method construction and (2) the *roadmap-driven* approach for method configuration in order to satisfy specific needs of each ISD crew member.

The remainder of this paper is as follows: in section 2 we propose a framework for SME combining the assembly-based and roadmap-driven approaches. The elements as method chunk and reuse frame shared by the two approaches are defined and illustrated in this section. Section 3 details and illustrates the assembly-based approach for project-specific method construction as first step of the global SME approach while section 4 deals with the road-map driven approach for method configuration as its second step. The conclusions and discussions about our future preoccupations are proposed in section 5.

2 Framework for Situational Method Engineering

In this section we present our framework for situational method engineering combining two complementary method engineering approaches called *assembly-based* and *roadmap-driven*. As shown in Fig. 1 illustrating our framework, the two approaches represent two core steps in the situation specific method construction-configuration process:

1. The aim of the first step is to build a new method the most adapted/suitable for the current ISD project situation. This step is found on the assembly-based method engineering approach detailed in section 3.
2. The second step aims at choosing the most adapted/suitable road-map within a selected method with regards to the project and ISD crew members needs. The roadmap-driven method configuration approach establishes the basis of this step. This approach is detailed in section 4 of this paper.

As shown in Fig. 1, both steps share two common elements: the *Method Chunks Repository* and the *Reuse Frame*. The former concerns the reusable method components definition and storage whereas the latter deals with the characterisation of the project situation and the definition of the requirements for a specific method or a road-map in a method. The method chunks repository contains the effectively reusable method knowledge defined in terms of method chunks, while the reuse frame contains the knowledge about the reuse context of these method chunks and provides criteria for project and project engineer situation characterisation.

Two core actors are identified in our framework (Fig. 1): the *Method Engineer* and the *ISD Crew Member*. Each of them has a specific role in the situational method engineering process. The method engineer is in charge of executing step 1 while the ISD crew member is responsible for the realisation of the step 2. In other words, the method engineer deals with the project specific method construction whereas the ISD crew member selects in this method the road-map corresponding to his or her tasks in the ISD project realisation.

Besides, the method engineer is responsible for filling up the method chunks repository, and the definition and management of reuse criteria in the reuse frame.

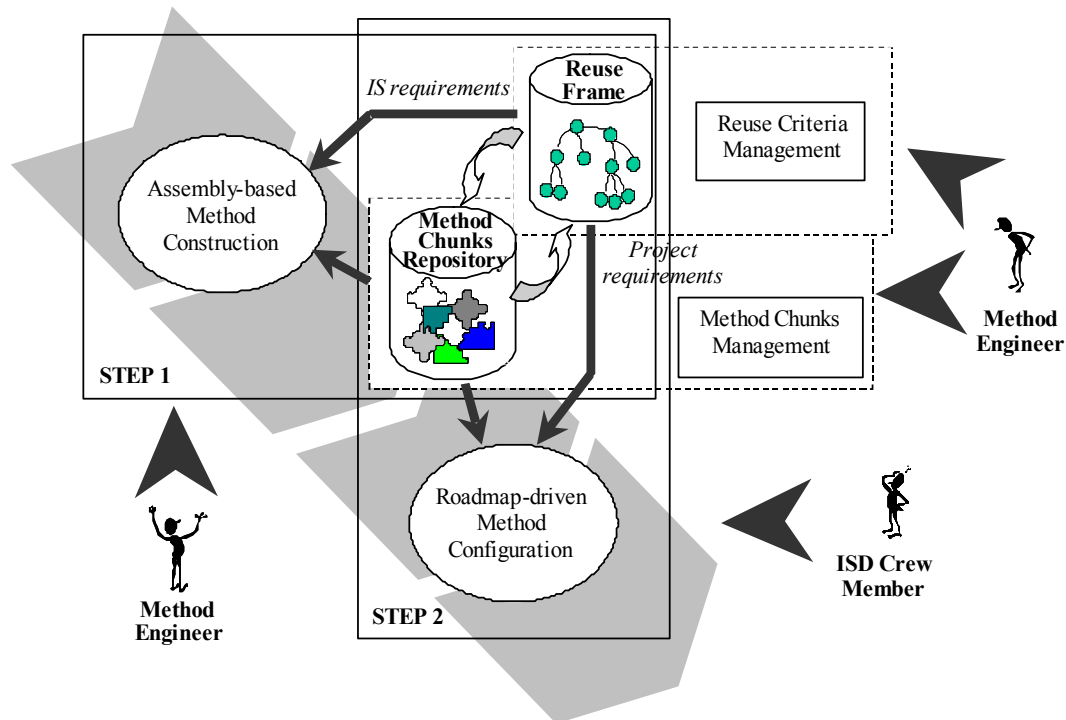


Fig. 1. A framework for situational method engineering combining assembly-based and roadmap-driven approaches

2.1 Method Chunks Repository

One of the core elements in our situational method engineering framework is the repository of method chunks (Fig. 1). The role of this repository is to store reusable parts of methods as prospective method building blocks.

Based on the observation that any method has two interrelated aspects, product and process, several authors propose two types of method components: process fragments and product fragments [Harmsen94, Brinkkemper98, Punter96]. Other authors consider only process aspects and provide *process components* [Firesmith02, Graham97], or *process fragments* [Mirbel04b, Mirbel02]. In our approach we integrate these two aspects in the same module that we call a *method chunk* [Plihon98, Rolland98, Ralyté01a]. The notion of *method bloc* proposed by Prakash in [Prakash99] is similar to the method chunk as it also combines product and process perspectives into the same modelling component. Both of these notions, *method fragment* and *method chunk*, represent the basic blocks for constructing ‘on the fly’ methods.

Van Slooten and Brinkkemper [Slooten93] combine method fragments into *route maps*. A complete route map represents a system development method. In our approach, the step 2 dealing with roadmap-driven method configuration also uses the notion of the route map to represent the specific path dedicated to a particular ISD crew member within a selected method [Mirbel04b].

Another kind of SME approaches uses generic *conceptual patterns* for method construction and extension [Rolland96a, Rolland96b] which capture generic laws governing the construction of different but similar methods. *Decision-making patterns* capturing the best practices in enterprise modelling are proposed by [Rolland00] to support enterprise knowledge development process.

Deneckere and Souveyet [Deneckere98] propose *domain-specific* process and product *patterns* for existing method extension.

In line with all these propositions around the notion of the reusable method component, we classify them into *method chunks*, *method fragments*, *patterns* and *road-maps* (Fig. 2).

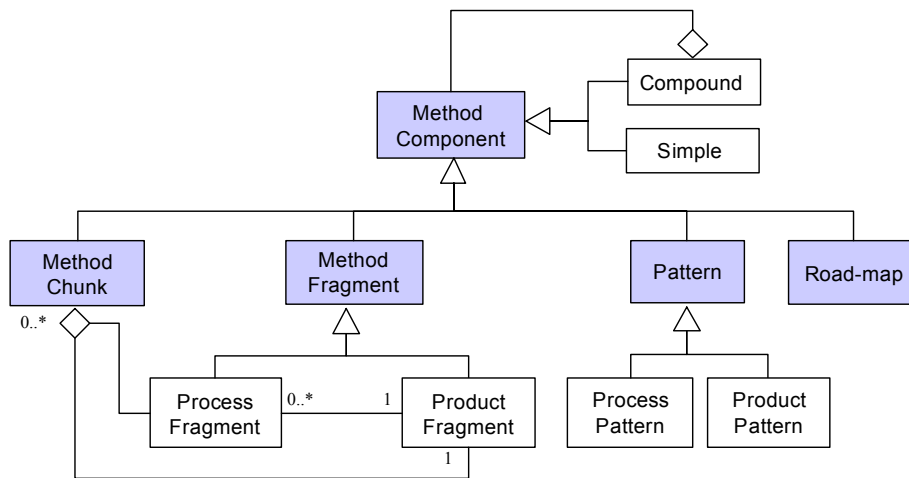


Fig. 2. Typology of reusable method components

In our approach we use two types of method components: method chunks and road-maps. A method chunk represents a reusable building block for situation-driven method construction or adaptation whereas a road-map represents a path in a method or a specific sequence of method chunks in a method.

2.1.1 Method Chunk

A method in our approach is viewed as a set of loosely coupled method chunks expressed at different levels of granularity. A method chunk is an autonomous and coherent part of a method supporting the realisation of some specific ISD activities. Such a modular view of methods favours their adaptation and extension. Moreover, this view permits to reuse chunks of a given method in the construction of new ones.

A method chunk ensures a tight coupling of some process part and its related product part. In the product part, the product to be delivered by the method chunk is captured whereas in the process part, the guidelines allowing to produce the product are given. For example, the method chunk providing guidelines (process part) for the use case model [Jacobson92] construction should also provide definitions of the concepts as actor, use case, scenario, extend relationship, etc. (product part) used in this model. As shown in Fig. 3 representing the notion of the method chunk, the corresponding product and process parts constitute the *body* of the method chunk.

The *interface* of the method chunk captures the reuse context in which the method chunk can be applied. It is formalised by a couple $\langle \textit{situation}, \textit{intention} \rangle$, which characterises the situation that is the input of the chunk process and the intention (the goal) that the chunk achieves. To *Construct a use case model* is an example of the intention associated to the method chunk providing guidelines for the use case model construction; the *Problem statement* specifies the situation in which this chunk can be applied that is the input product necessary to start the execution of this method chunk.

Besides, a *descriptor* is associated to every method chunk. It extends the contextual view captured in the chunk interface to define the context in which the chunk can be reused. The descriptor takes values from the reuse frame and associates them to the corresponding method chunk.

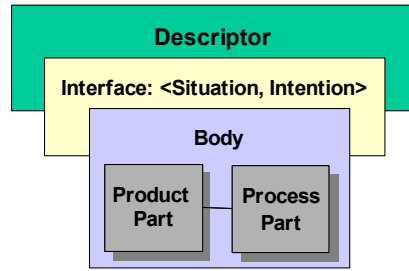


Fig. 3. Notion of the method chunk

Fig. 4 shows our meta-model for modular methods. According to this meta-model, a method is also viewed as a method chunk of the highest level of granularity. The body of the method chunk captured a part of method process model called *guideline* that can be considered as autonomous and reusable and a part of its product model needed to perform the process encapsulated in this guideline.

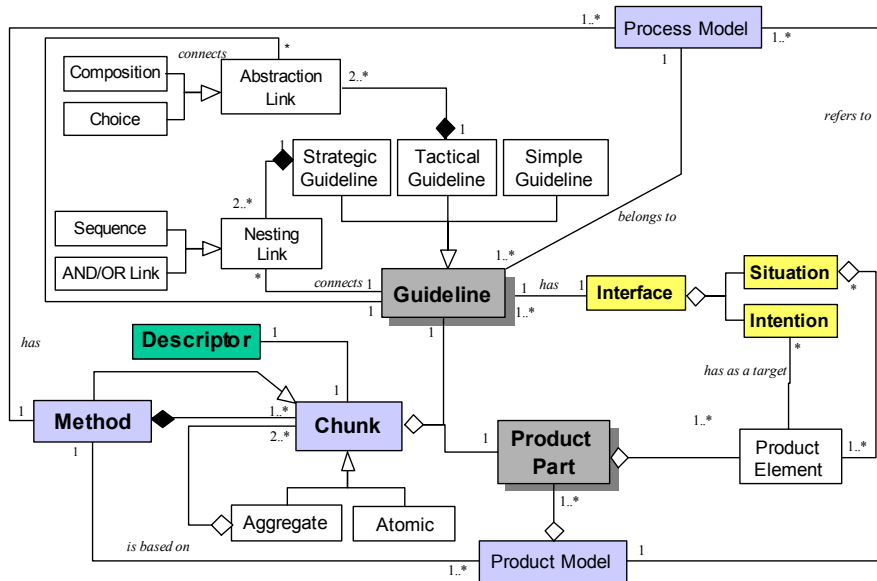


Fig. 4. The meta-model for modular methods

A guideline is defined as ‘a statement or other indication of policy or procedure by which to determine a course of action’ [AHD00]. For us, a guideline embodies *method knowledge* to guide the application engineer in achieving an intention in a given situation. Therefore, the guideline has an *interface*, which describes the conditions of its applicability (the situation) and a *body* providing guidance to achieve the intention, i.e. to proceed in the construction of the target product. As mentioned above, the interface is a couple $\langle \textit{situation}, \textit{intention} \rangle$. A situation represents an input to the method chunk process and contains one or several *product elements* necessary to start its execution in order to achieve its intention. An intention represents an engineering goal to be achieved by applying the method chunk. An intention is expressed following a linguistic approach proposed by Prat [Prat97] as a clause with a *verb* and a *target*. 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 use case model following OOSE method advices*, the verb ‘Construct’ is followed by the target product ‘a use case model’ and the manner to achieve this intention ‘following OOSE method advices’. It is formalised as follows: $\textit{Construct}_{Verb} (\textit{a use case model})_{Target} (\textit{following OOSE method advices})_{Manner}$.

The product elements used in the method chunks situations as well as the verbs and target products specified in the method chunks intentions are defined in our method engineering glossary. They are used during the retrieval process of the method chunks from the repository.

The body of the guideline details how to apply the chunk to achieve its intention. The interface of the guideline is also the interface of the corresponding method chunk. Guidelines in different methods have different contents, formality, granularity, etc. In order to capture this variety, we identify three types of guidelines: simple, tactical and strategic.

A *simple guideline* may have an informal content advising on how to proceed to handle the situation in a narrative form. It can be more structured comprising an *executable* plan of actions leading to some transformation of the product under construction. Fig. 5 illustrates a method chunk with a simple informal guideline capturing the advices proposed by the OOSE method to support the identification of the actors of the system under construction by asking a set of typical questions. The product part of this chunk defines the concept *Actor*.

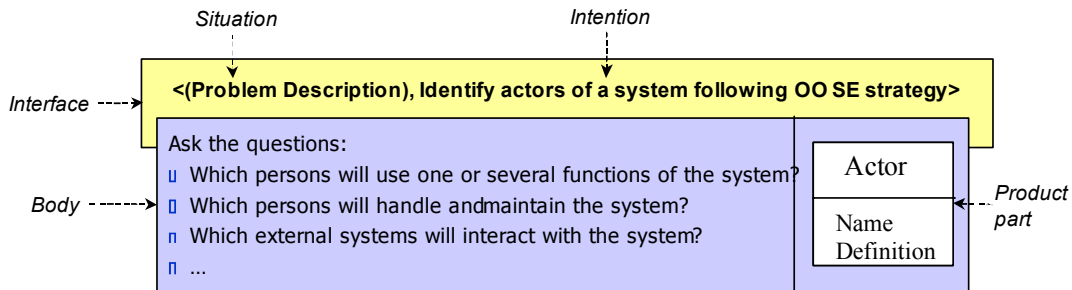


Fig. 5. Example of a method chunk with a simple informal guideline

A *tactical guideline* is a complex guideline, which uses a tree structure to relate its sub-guidelines one with the others. This guideline follows the *NATURE* process modelling formalism [Jarke99], which proposes two different structures: the *choice* and the *plan*. Each of its sub-guidelines belongs to one of these types of guideline. Fig. 6 illustrates a method chunk with a tactical guideline for ER schema construction. This guideline is defined as a plan composed of three sub-guidelines: **<(Problem Statement), Define Entity-Type ET>***, **<(Problem Statement, {ET}), Define Relationship-Type RT>*** and **<(ER schema), Refine ER schema>***. The 'star' mark next to the guideline definition means that the corresponding guideline can be repeated N times. Moreover, each of these sub-guidelines is defined as a plan or a choice of several sub-guidelines.

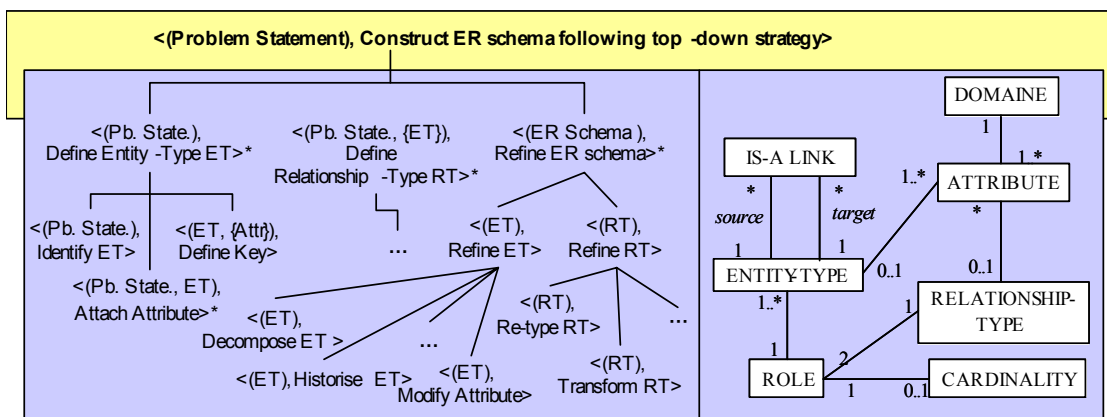


Fig. 6. Example of a method chunk with a tactical guideline

A *strategic guideline* is a complex guideline called a *map* [Rolland99] that uses a graph structure to relate its sub-guidelines. Each sub-guideline belongs to one of the three types of guidelines. A *strategic guideline* provides a strategic view of the development process telling which *intention* can be achieved following which *strategy*. An *Intention* is a goal that can be achieved by the performance of an activity (automated/semi-automated or manual). There are two special intentions *Start* and *Stop* that allow to begin and to end the progression in the map, respectively. An intention is expressed following

a linguistic approach mentioned above [Prat97]. A *Strategy* is an approach, a manner to achieve an intention. Thus, a map is a labelled directed graph in which the nodes are the intentions and the edges between intentions are strategies.

The map permits to represent a process allowing several different ways to develop the product. A set of guidelines is associated to the map. They help the application engineer to progress in the map and to achieve the intentions following selected strategies. More exactly, a map is a composition of a set of sections where a section is a triplet $\langle \text{Source intention}, \text{Target intention}, \text{Strategy} \rangle$. Every section provides an *Intention Achievement Guideline* (IAG) indicating how to achieve the target intention following the strategy given the source intention has been achieved. Two other types of guidelines, *Intention Selection Guideline* (ISG) and *Strategy Selection Guideline* (SSG), help to progress in the map i.e. to select the next intention and to select the next section respectively.

To illustrate the strategic guideline we propose to consider the *L'Ecritoire* approach [Rolland98a, Rolland98b, Tawbi98]. This method chunk provides guidelines to discover functional system requirements expressed as goals and to conceptualise these requirements as scenarios describing how the system satisfies the achievement of these goals. As shown in Fig. 7, the corresponding guideline is represented by a map with three core intentions: *Elicit a goal*, *Write a scenario* and *Conceptualise a scenario*. Several different strategies are provided by the chunk to achieve these intentions each of them representing a different manner to do it.

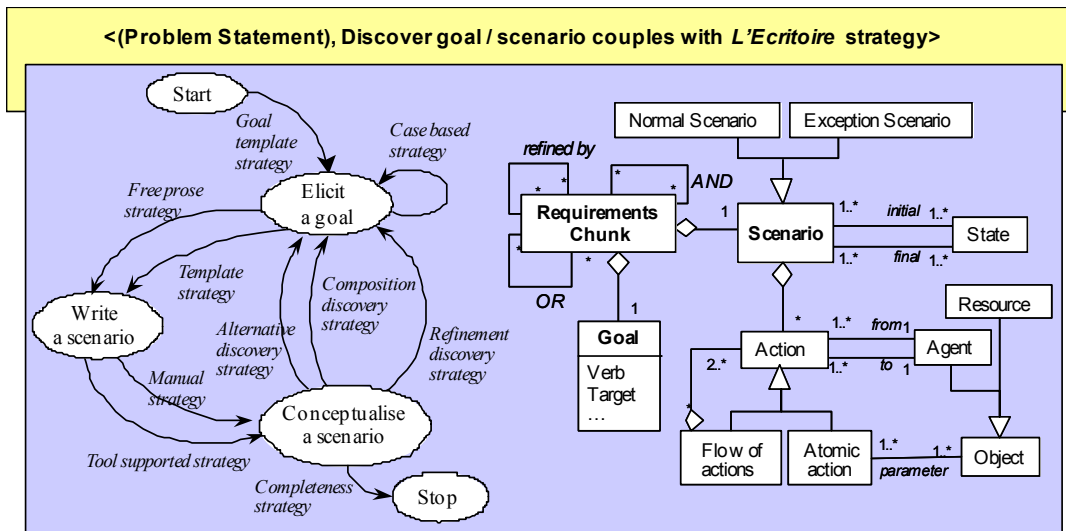


Fig. 7. Example of a method chunk with a strategic guideline

As mentioned above, each method chunk has a *Descriptor* which captures the knowledge about its reuse conditions. In other words, the knowledge contained in the descriptors are used during the method chunks selection and retrieval from the repository. As shown in Fig. 8, the two key elements of the descriptor are the *reuse context* and the *reuse intention*. The reuse context captures a set of criteria taken from the reuse frame and characterising the corresponding method chunk. For example, the reuse context of the method chunk presented in Fig. 7 should specify that this chunk is applicable during the requirements elicitation activity. In the next section we provide more details about the reuse frame.

The reuse intention expresses the objective that the method chunk helps to satisfy in the corresponding engineering activity. The reuse intention has the same structure as the chunk intention that is defined by a verb, a target and a set of parameters which are specified in the glossary. For example, the reuse intention of the method chunk presented in Fig. 7 is defined as $Discover_{verb}(\text{functional system requirements})_{Target}(\text{following } L'Ecritoire \text{ strategy})_{Manner}$.

Besides, the descriptor contains the *name* and the *ID* of the corresponding method chunk and a narrative description of its *objective*. It also specifies the *type* (i.e. atomic or aggregate) and identifies the *origin* of the chunk (i.e. the method from which this chunk is extracted). If the method chunk is an

aggregate one, its descriptor specifies its components and vice versa, if the chunk takes part of some aggregate chunks its descriptor identifies them. The *Aggregates/Components* relationship allows to navigate from one descriptor to another during the selection process without looking at the content of the corresponding chunks. The *alternative* and *not-compatible* method chunks to the given one can also be specified in the descriptor indicating the degree of complementarity and incompatibility respectively. Some application *examples* and *experience* reports can be added in order to facilitate method chunks selection in the case where several chunks would be retrieved to satisfy the same intention.

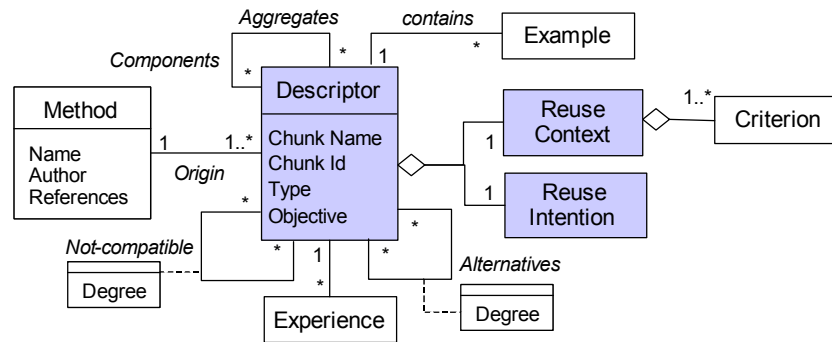


Fig. 8. The descriptor

2.1.2 Road-map

A road-map is composed of one or several coherent sequence(s) of method chunks taken from a situational method described in terms of chunks and corresponding to the usage of this method by a particular ISD crew member. It is a customized view of a situational method to answer the specific need of an ISD crew member. We distinguish 3 categories of customized view of a situational method:

- The ISD crew member needs help on a specific step of the ISD method: for instance, he/she looks for the method chunks dealing with the requirement elicitation activity. In such a case, a set of organized chunks covering specific required stages of the ISD process is provided.
- The ISD crew member asks for help on a specific point of the ISD method: for instance he/she looks for the guidelines to build a class diagram by using the UML notation. In this case, a small set of chunks, not related among them, is provided.
- The ISD crew member wishes the situational method to be presented from a specific view point: for instance, he/she looks at guidelines related to the fact that the ISD takes place on top of a legacy application. In this case, a large set of chunks covering as much as possible the ISD process, with regards to the chosen view point, is provided.

Through the road-map notion, our goal is not to give ISD crew members the full and detailed description of all the tasks they will have to deal with during ISD because spaces of creativity are also required, but to give them advices when they ask for it and/or when guidelines about similar situation have been accumulated in the repository. Road-maps are used in the second step of our approach dealing with the configuration of a situational method for an ISD crew member. Our purpose in this step is not to provide another way to build a method but to propose different ways to make the situation-specific method usable and useful 'on the fly' for a specific ISD crew member.

2.2 Reuse Frame

Looking at the way ISD methods are used in practice, we notice they are always adapted: steps are added, other removed or skipped and so on. Different factors related to the project, the technology, the team expertise and the business domain lead to this tailoring. Method tailoring is supported by the assembly of predefined method chunks [Brinkkemper98, Rallyté01a]. Dedicated efforts have been

made, in the field of method engineering, to provide efficient classification and retrieving techniques to store and retrieve method chunks. Classification and retrieving techniques are currently based on structural relationships among chunks (specialization, composition, alternative, etc.) and reuse intention matching.

From our point of view, current classification and retrieving means do not fully exploit the potential of breaking down ISD methods into method chunks and tailoring them. Therefore, we propose the notion of *reuse frame* that we detail in the following sections.

2.2.1 A reuse frame to handle critical aspects of information systems

We believe that knowledge about organizational, technical and human factors, which is critical knowledge about ISD [Cauvet01], should be taken into consideration in addition to structural knowledge and reuse intention. It will allow to better qualify method chunks when entering them into the repository and to enable the use of more powerful matching techniques to find them again when looking at similar ISD methodological problems. It will also allow to better express methodological needs for a specific ISD project, improving this way the chance to get adequate and useful method chunks.

Therefore, we propose a *reuse frame* aggregating the different ISD critical aspects useful to tailor ISD methods with regards to organizational, technical and human dimensions of Information Systems (IS). Of course, method chunks, as well as methodological needs, may be defined more or less precisely with regards to ISD critical aspects. Therefore, these three aspects have to be refined and presented in a way supporting the required polymorphism, which is the case of our reuse frame which is a tree of successively refined aspects. By providing such an ontological structure, we enlarge the panel of means for method engineers to broadcast ways of working which is most of the time reduced to deliverables. And we provide to ISD crew members additional information to find the most suitable information with regards to his/her ISD methodological need (or problem).

ISD critical knowledge is described in terms of *aspect*, belonging to *aspect families*, which are successive refinements of the three main factors of IS: human, organizational and technical. With regards to the *organizational* dimension, we started from the work of van Slooten and Hodes providing elements to characterize ISD projects [Slooten96]. With regards to the *technical* dimension, we started from previous work on *JECKO*, a context-driven approach to software development, including a contribution to define software critical aspects in order to get suitable documentation to support software development process [Mirbel02, Mirbel03]. The technical dimension also includes aspects related to application domain and system engineering activities. And finally, about the *human* dimension, we currently propose a basic description of the expertise of the ISD crew members (low, medium or high).

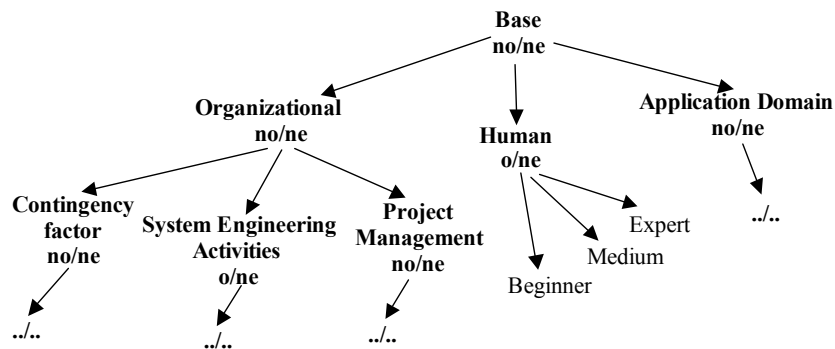


Fig. 9. Reuse frame

All the knowledge about ISD critical aspects is included into the reuse frame taking form of a tree, where leaf-nodes are aspects and intermediary nodes are families. Indeed, nodes close to the root node are seen as general aspects while nodes close to leaf nodes (including leaf-nodes) are seen as precise aspects. The top of the reuse frame is shown in Fig. 9. Refer to the annex for detail of each branch.

A leaf-node (or aspect) is defined through a *name*. An intermediary node (or family) is also defined through a *name* and completed by information about relationships among the different aspects or sub-families belonging to it. This additional information, aiming at better specifying the way aspects belong to a family, helps in using them to constitute a coherent context. Two kinds of information are provided:

- The *order* field indicates if direct aspect or sub-families are ordered (*ord=o*) or not (*ord=no*). For instance, if *system engineering activity* is a *family*, *analysis*, *design*, *development*, *deployment* and *test* are aspects which are ordered (*analysis* is processed before *design*, which is processed before *development*, which is processed before *deployment*, which is processed before *test*). It is interesting to indicate this information because when retrieving fragments associated with the *design* aspect for instance, it may also be interesting to look at fragments associated with the *analysis* and *development* aspects, especially if one is interested in the beginning and ending steps of the *design* phase.
- The *exclusion* field indicates if direct aspects or sub-families are exclusive (*exc=e*) or not (*exc=ne*). For instance, there is in the reuse frame an aspect related to project time pressure. Guidelines may be given for projects under high time pressure as well as projects under low time pressure. Therefore *time pressure* is a *family* and *low time pressure* and *high time pressure* are aspects. We specify them as exclusive aspects because guidelines associated to high time pressure projects are not compatible with guidelines associated with low time pressure projects and could not be provided in the same project-specific method or roadmap.

Indeed, the **Reuse Frame**, *RF*, is a tree where:

- the root node is defined as $\langle \text{name=base, exc=ne, ord=no, type=root} \rangle$,
- non-leaf nodes are family defined as $\langle \text{name, exc, ord, type= family} \rangle$,
- leaf nodes are aspects defined as $\langle \text{name, type=aspect} \rangle$

where *exc= e* or *ne* respectively for exclusive or non-exclusive and *ord=o* or *no* for ordered or non-ordered.

As critical aspects of ISD may evolve through time, means have to be provided to make the reuse frame content evolve too. Method engineers could make the reuse frame evolve by specifying more deeply existing aspects by refining them (i.e. transforming aspects into families grouping new aspects) or by reorganizing existing families by adding new intermediary nodes.

The reuse frame allows method engineers to drive the ISD crew members to focus on critical aspect(s) of ISD whatever the method evolution is and this way ensure to always take as much advantage as possible from the method fragmentation and tailoring mechanisms, as it has been previously presented as one of our main goals.

2.2.2 Taking advantage of the reuse frame through ISD

As it has been introduced before, the reuse frame is used (i) when inserting new method chunks into the repository in order to improve their specification with regards to ISD features and enabling more powerful matching techniques for their retrieval and (ii) when searching for process chunks to answer a specific methodological need and to retrieve adequate method chunks in order to build a road-map dedicated to a specific ISD crew member.

Therefore, means have to be provided to select from the reuse frame the pertinent aspects associated to chunks and methodological need/problem (and its solutions). It is done through the notion of *criterion* and *context*.

A **criterion** is fully defined as a path from the root node *base* to a node n_n of the reuse frame.

$$Cr = [base, n_1, \dots, n_n] \text{ with } base, n_1, \dots, n_n \in CF$$

If n_n is a *family* node, then the exclusion field *exc* must be different from *e* because one of its *aspects* has to be chosen inside the *family*.

if type_n = family, exc_n ≠ e

A **context** is defined as a set of *compatible* criteria.

$$Co = \{Cr_1, \dots, C_m\}, \forall Cr_i, Cr_j \in Co, Cr_i \text{ comp } Cr_j$$

Two criteria are **compatible** if they do not share in their definition a common *family* node n_i with an *exclusion* field equal to e .

$$Cr_1 \text{ comp } Cr_2, \forall n_i \in Cr_1 \text{ and } n_j \in Cr_2, \text{ if } n_i = n_j \text{ then } exc_{n_i} \neq e$$

The **Method Chunk Reuse Context** is defined as a set of at least one compatible criterion taken from the *Reuse Frame*.

$$MCC = \{C_1, \dots, C_n\}, \forall C_i \in MCC, C_i \in RF, \forall C_i, C_j \in MCC, C_i \text{ comp } C_j$$

Method chunks providing general guidelines are usually associated to general criteria, that is to say paths ended by nodes from the reuse frame close to the root node. On the contrary, specific guidelines are provided in method chunks associated to precise criteria, that is to say paths ended by nodes from the Reuse Frame close to aspect nodes or aspect nodes themselves.

3 Step 1: Assembly-based Method Construction

The approach for assembly-based method construction aims at building a method ‘on the fly’ in order to match as well as possible the situation of the project at hand. It consists in selecting reusable method components (that we call method chunks) according to the current project requirements from the method chunks repository and assembling them.

Fig. 10 summarizes our approach for assembly-based method construction. This approach is requirements-driven, meaning that the project method engineer must start by eliciting requirements for the project method. Next, the method chunks matching these requirements can be retrieved from the method chunks repository. And finally, the selected chunks are assembled in order to compose a new method or to enhance an existing one. As a consequence, the three key steps in the assembly-based method engineering process are: *Specify Method Requirements*, *Select Method Chunks* and *Assemble Method Chunks*. Our approach provides guidelines supporting method engineer in each of these steps.

The *requirements specification guidelines* help to define the requirements for the project specific method in terms of required IS development activities by specifying the type of these activities and the order of their execution. Goal-driven requirements elicitation, scenario-based requirements specification and negotiation, conceptual schema definition, interface modelling are the examples of such activities. The reuse frame is used in order to characterize project situation and to select the predetermined activities.

The *chunk selection guidelines* advise method engineer how to retrieve method chunks from the method repository in order to satisfy requirements identified in the previous step. The reuse context of each method chunk is characterized by a set of criteria defined in the reuse frame. Therefore, the reuse frame helps to retrieve method chunks from the method repository.

Finally, the *method construction guidelines* assist method engineer in the selected method chunks assembly process in order to obtain a homogeneous method. We detail these three steps in the following sub-sections.

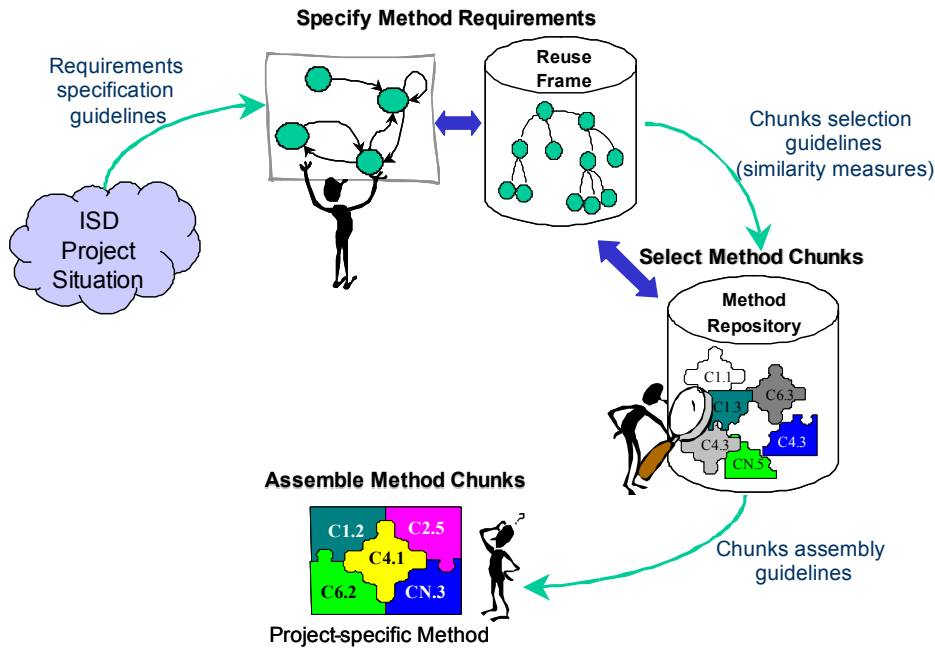


Fig. 10. Overview of the approach for assembly-based method construction

3.1 Method Requirements Specification

The specification of method requirements for a particular ISD project depends on its initial method situation and the corresponding Method Engineering (ME) goal. Two core situations can be identified:

1. The IS development crew is used to follow the same method for all ISD projects but considers that in the current project situations this method must be adapted.
2. The IS development crew does not possess any regular method.

In the first case, the ME goal is to *adapt an existing method* whereas in the second case a brand new project-specific method must be constructed.

There are different types of method adaptation. In this approach we distinguish three of them:

- the method in use can be strong in terms of its product model but weak with respect to its process model, which will be the subject of adaptation and enhancement, alternative ways-of-working will be added in the method to its original one;
- the adaptation can be to simply add a new functionality to the existing method, which is relevant in its other aspects;
- vice versa, the project at hand could not need some functionality offered by the method.

In all these cases, the requirements elicitation process is driven by the identification of the ME intentions such as ‘add event concept’, ‘complete completeness checking step’, ‘replace scenario writing guidelines by the more rich ones’ etc., which will allow to complete, enhance or limit the method initially selected. The three adaptation cases can be combined in the selected method adaptation.

In the case of a brand new method construction, the requirements specification is not only to produce the list of ME intentions that will permit to adapt the selected method but to identify the full set of engineering intentions that shall be fulfilled by the new method.

Both of these requirements specification cases lead to a set of requirements expressed as a map (i.e. strategic guideline) [Rolland99] that we call the *requirements map* [Ralyté02]. Let us suppose that a

method engineer has to construct a new method supporting: (1) the elicitation of functional system requirements in a goal-driven manner, (2) their conceptualization by using textual devices such as scenarios or use cases and (3) their validation in an animated fashion as prototyping and other simulation mechanisms. The requirements map represented in Fig. 11 captures these requirements in three core intentions: Elicit a requirement, Conceptualise a requirement and Validate a requirement and imagines several possible strategies to achieve these intentions.

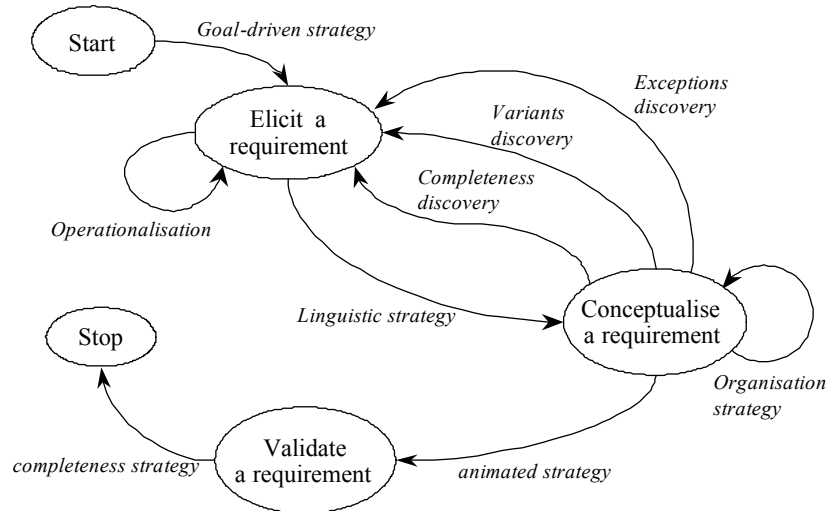


Fig. 11. An example of a requirements map

3.2 Method Chunks Selection

Once the method requirements have been specified, the selection of the method chunks matching these requirements can start. The selection process is based on the requirements map defined in the previous step. The objective that the method engineer has to attain in this step is to select method chunks each satisfying a part of requirements that is covering at least one strategy of the requirements map. Otherwise, a selected method chunk can satisfy all the requirements and cover the whole requirements map.

We formulate the chunk selection queries by specifying the reuse intention of the method chunk and giving values to different reuse context criteria specified in the chunk descriptor [Plihon98, Ralyté99a,b] as, for example, the system engineering activity in which the chunk is relevant. The situation and intention defined in the chunk interface can also be useful during the chunk selection process. For example, if we need a method chunk supporting requirements elicitation and specification from initial problem statement, and specifying requirements in terms of goals and/or scenarios we should put these values in the query as follows:

```

Reuse_intention.verb = 'Discover' AND Reuse_intention.target =
'Functional system requirements' AND Reuse_context.Engineering
activity = ('Requirements elicitation' AND 'Requirements
specification') AND Situation = 'Problem statement' AND
Intention.Target = ('Goal' OR 'Scenario')

```

Each retrieved chunk is validated by evaluating its degree of matching to the requirements. This is done by applying similarity measures between the requirements model and the process model of the selected chunk. More details about these similarity measures could be found in [Ralyté01a]. For example, the method chunk provided in Fig. 7 partly satisfies the requirements captured in the requirements map. It provides guidelines for functional system requirements elicitation in form of goal hierarchies and requirements conceptualisation in form of scenarios.

The chunk selection process also provides different guidelines allowing to refine the candidate chunk selection by analysing more in depth if the chunk matches the requirements. For example, if the selected method chunk is an aggregate one, i.e. it is composed of several chunks, the *Decomposition guideline* drives the selection of the relevant sub-chunks and the elimination of the inadequate ones. Vice-versa, if the retrieved chunk matches partly the requirements, the *Aggregation guideline* proposes to look for an aggregate chunk containing the candidate one based on the assumption that the aggregate chunk might provide a solution for the missing requirements. It is also possible to refine the chunk selection query by modifying selection parameters.

3.3 Method Chunks Assembly

When at least two chunks have been selected, the method engineer can progress in the assembly of these chunks. We have identified two core types of guidelines in the method chunks assembly process that we call *assembly by association* and *assembly by integration* [Ralyté01a].

The *Assembly by association* is relevant when the method chunks to assemble correspond to two different system engineering functionalities, i.e. they allow to achieve different engineering intentions and the result of one chunk is used as a source product by the second one. Such method chunks generally do not have common elements in their product and process models and the assembly process is therefore mainly dealing with making the bridge between the two chunks. More precisely, the product models of the chunks must be connected by defining links between their different concepts whereas the connection of their process models consists in defining their execution order.

For example, the method chunk providing guidelines for object life cycle definition requires that the corresponding object class and its relationships with other classes have been identified beforehand. Some information about the events that could occur during the systems life cycle and have some impact on this object class should also be provided. As a consequence, the method chunks for object model construction and events definition must be realised before the chunk for object life cycle definition. The three chunks have complementary objectives. Their assembly is limited to the connection of corresponding concepts as *class*, *state* and *event* and the identification of the order in which they must be executed.

The *Assembly by integration* is relevant to assemble chunks that have similar engineering goals but provide different ways to satisfy it. In such a case, the process and product models are overlapping, that is containing some identical or similar elements. The assembly process consists in identifying the common elements in the chunks product and process models and merging them. For example, it could be useful to assemble two method chunks providing different guidelines for requirements elicitation. The chunk supporting goal-driven requirements elicitation combined with the one dealing with actor-driven requirements elicitation allow us to obtain a new chunk providing guidelines more rich than the two initial chunks used separately.

Both types of guidelines use a set of assembly operators, operators for product models assembly and operators for process models assembly [Ralyté99a, Ralyté04].

For example, the association of two product models is mainly based on the *AddAssociation* and *AddClass* operators: a new association is created between the classes of the initial models in order to connect them into a new model, it can also be done by adding a new class and two associations. On the contrary, the integration of two product models is based on the application of the *Merge* operators as *MergeClasses*, *MergeAssociations* which allow to connect the initial models by merging similar or identical classes and associations respectively. *ConnectViaSpecialization* and *ConnectViaGeneralization* operators define respectively the specialisation and generalisation links between the concepts of the different product models. Their application is useful to build a model of the integrated method chunk that is richer than those of the initial chunks.

In the same manner, the process models of the selected method chunks are assembled: a new transition between two activities can be added in order to connect two activity-driven models or two activities can be merged into a new one in the case of the overlapping process models integration. If the chunks process models are expressed by using map formalism [Rolland99], the *MergeIntentions* and *MergeSections* operators are applied.

Some method chunks adaptation could be required before their assembly in order to avoid concepts name ambiguity. This may be done by applying different *Rename* operators as *RenameClass*, *RenameAttribute*, *RenameActivity*, etc., as well as *Retype* operators as *RetypeClass*, *RetypeAttribute*, *RetypeIntention* etc.

The result of the assembly process is a new method that we call here a *project-specific method*. This method is an instance of our meta-model for modular methods (Fig. 4). Therefore, it is a collection of tightly coupled method chunks. More exactly, the guideline of this method is a strategic one where each section of its map corresponds to a method chunk or a part of a method chunk.

4 Step 2: Roadmap-driven Method Configuration

The approach for roadmap-driven method configuration aims at customizing the project-specific method 'on the fly' in order to match as well as possible the profile of the ISD crew member job within the project at hand. It consists in the selection of method chunks satisfying the ISD crew member needs within the frame of the project-specific method previously built to answer project requirements.

Indeed customisation of ISD methods have mainly be thought for the person in charge of building new methods, i.e. the method engineer, in order to allow him/her to construct or adapt a method satisfying the needs of his/her company or the requirements of a specific ISD project. But there is also a need for customisation dedicated to each ISD crew member, to provide him or her with guidelines to be followed while performing their daily tasks. ISD crew members also need to benefit, through reuse and adaptation mechanisms, from the experiences acquired during the resolution of previous problems in terms of applying methods in real ISD projects (i.e. using it).

As it has been highlighted during the assembly-based method construction step, different ways may be provided, even inside the same method, to satisfy an engineering goal. Moreover, the sequence through which method chunks have to be used is not always pre-determined: they may or not be related by precedence relationships. Therefore, different road-maps among a set of method chunks are possible. Through this second step of our approach for road-map driven method configuration we provide different means to find the road-map which is the most suitable for the needs of a particular ISD crew member.

The road-map driven method configuration is based on three main steps and their underlying techniques: (1) the specification of the methodological problem of the ISD crew member, (2) the retrieval of method chunks matching the problem from the project-specific method built through the first step of our approach and/or from the method chunks repository, and (3) the completion of the solution by additional method chunks retrieved from the repository.

In the following we first detail each of these techniques and then we discuss the different possible ways to use them in order to get a suitable solution to the methodological problem of an ISD crew member.

4.1 Techniques

4.1.1 Specifying the Problem of the ISD Crew Member

First of all, the ISD project crew member must express its problem in terms of applying the project-specific method (resulting from the first step of our approach). The problem is specified through a *problem context* and eventually a problem intention that the selected method chunks should match. In the problem context, in addition to the pertinent criteria, called *necessary criteria*, one may need to give *forbidden criteria*, that is to say aspects the ISD project crew member is not interested in. It could be helpful in some cases to be sure the method chunks including these (forbidden) aspects will not appear in the solution.

A **Problem Context** is defined as CN , a set of at least one compatible *necessary criteria* and CF , a set of compatible *forbidden criteria*. All criteria are taken from the reuse frame.

$$PC = \langle CN, CF \rangle \text{ where } CN = \{C_1, \dots, C_n\}, CF = \{C_1, \dots, C_m\}$$

$$\forall C_i \in CN, \neg \forall C_i \in CF$$

$$\forall C_i \in CN, C_i \in RF, \forall C_i \in CF, C_i \in RF$$

$$\forall C_i, C_j \in CN, C_i \text{ comp } C_j, \forall C_i, C_j \in CF, C_i \text{ comp } C_j$$

4.1.2 Selecting Method Chunks in the Project-specific Method

Thanks to the method chunk descriptor, the selection of the method chunks from the project-specific method the ISD crew member may be interested in can be done by matching problem context and method chunk reuse context, as well as problem intention and method chunk reuse intention. Glossaries are provided to specify the intention through meaningful set of verbs, targets and parameters. We introduce the situational metric to quantify the matching between method chunk reuse context and problem context [Mirbel04b]. This metrics is based on (i) the number of common criteria between the necessary criteria (from the problem context) and the method chunk reuse context, (ii) the number of common criteria between the forbidden criteria (from the problem context) and the method chunk reuse context, (iii) the number of required necessary criteria (from the problem context). A positive value indicates that there are more necessary criteria than forbidden ones in the method chunk under consideration with regards to the problem. On the contrary, a negative value indicates that there are less necessary criteria than forbidden ones. The perfect matching is represented by the value 1.

Thanks to this metrics, used in addition to problem intention matching, a set of method chunks corresponding to the ISD project crew member profile is obtained as result of the search. Indeed, the selected method chunks may not be related all together: they may for instance cover two disjoint stages of the method (at the beginning and the end of the method for instance) i.e. two disconnected sections in the project-specific method map. So, the result set is in fact made of different subsets of organized method chunks.

4.1.3 Searching for Method Chunks in the Repository

Method chunks could also be searched directly inside the repository using the same techniques: reuse intention matching and situational metrics. It could be useful by itself to search for chunks directly in the repository without looking at the project-specific method, as it will be explained in section 4.2.

In the repository, method chunks may be related to each other as *alternative method chunks* or *not compatible method chunks*. Degrees are also associated to these kinds of association to respectively quantify the complementarity and the incompatibility. When building a road-map, the coherency of the selected method chunks has to be enforced by fulfilling the constraints expressed through these relationships. Such a check was not necessary when selecting method chunk from the project-specific method, because consistency among the different method chunks was ensured by the project-specific method. When selecting method chunks directly from the repository, complementary method chunks with high degrees have to be added to the solution and not compatible method chunks with high degrees have to be removed from the solution. Indeed, the ISD project crew member gives alternativity and incompatibility thresholds to tune the level of coherency he/she wants his/her road-map to satisfy. A high alternativity threshold will lead to a solution in which only very alternative method chunks will be added, while a low one will lead to a solution in which most of them will be included. A high incompatibility threshold will lead to the removal of the very incompatible method chunks from the solution, while a low threshold will lead to the removal of most of the incompatible method chunks.

Indeed, a road-map may be constituted of different organized sequences of method chunks. Therefore, if in a sequence, a method chunk appears in the list of not compatible method chunks associated to another method chunk with a degree higher than the incompatibility threshold, then the current method chunk is removed from the road-map.

The ISD crew member can also look for method chunks directly in the repository in order to complete the set of chunks embedded in the project-specific method. In this case, the search in the repository is valid but it has to be completed; method chunks declared as incompatible with those assembled in the project-specific method have to be removed from the road-map.

Finally, the set of method chunks still kept in each sequence constitutes the road-map corresponding to the methodological need of the ISD project crew member.

4.1.4 Tuning Facilities

The reuse frame introduced previously supports different levels of granularity with regards to criterion definition and use. Criteria with ending node close to the *base* nodes are much more generic than criteria with ending node close to the *aspect* nodes. Indeed, method chunks providing general guidelines are usually associated to general criteria, while specific guidelines are provided in method chunks associated to precise criteria.

When searching for method chunks, one may be interested by method chunks having criteria strictly matching the necessary criteria defined in the problem context. But method chunks with reuse context associated to more specific criteria and usually providing more specific guidelines may also be of interest. They may for instance cover a bigger part of the methodological problem the ISD crew member has to deal with. In the same way, the ISD crew member may be interested in method chunks having reuse context associated to more general criteria and thus providing more general-purpose guidelines which could also be relevant.

Taking into consideration method chunks with reuse contexts containing more general and/or more specific criteria may also be interesting with regards to the forbidden criteria given in the problem definition. Indeed, enlarging the set of forbidden criteria to more general ones means to forbid full branches of the reuse frame, while enlarging the set of forbidden criteria to more specific ones means to forbid method chunks associated to too specific criteria.

Tuning the selection by allowing or not more general and/or more specific criteria to be included in the necessary and/or forbidden criteria given in the problem definition provide a way for the ISD crew member to reduce or enlarge the number of method chunks included in the solution of his/her problem. If the ISD crew member feels that he/she did not retrieve enough method chunks, he/she may allow more general and/or more specific criteria in order to find more method chunks. On the contrary, if the set of method chunks provided as an answer to his/her problem is too big, he/she may enlarge the set of forbidden criteria by allowing more general and/or more specific criteria and this way cutting branches of the reuse frame and therefore removing from the solution the method chunks associated to these criteria.

4.2 Road-map Building Approaches

Based on the techniques we introduced (problem specification, method chunk search, selection, and tuning) we propose different ways to build a road-map.

The most obvious way to build a road-map consists in selecting the suitable method chunks inside the project-specific method obtained by applying the first step of our approach. Such a way to build road-maps is mainly dedicated to neophytes or ISD crew members who want to follow carefully the method built for the project purposes. We call this way to construct road-maps a *guided road-map building*.

A much more *free* way consists in searching method chunks directly in the repository without looking at the project method map. Such a way to build road-maps may be recommended when the ISD crew member is searching for an answer to a very specific problem (leading to the selection of 2 or 3 specific chunks).

And finally, a *balanced* way consists in selecting method chunks from the project-specific method and then completing the set of selected chunks by additional ones taken from the repository. Additional chunks could be relevant if they represent alternatives to the chunks embedded in the

project-specific method built through the first step of our approach or if they fully answer the problem intention and context given by the project team member. After the application of the chunks from the project-specific method in combination with the additional chunks, the ISD crew member can then give a feed-back to the method engineer about his/her experience in applying these additional chunks and ask him/her to assemble some of them into the project method if he/she feels it could be useful for other ISD crew members of this project.

In the following, we detail each of these three ways to build road-maps summarized in Fig. 12.

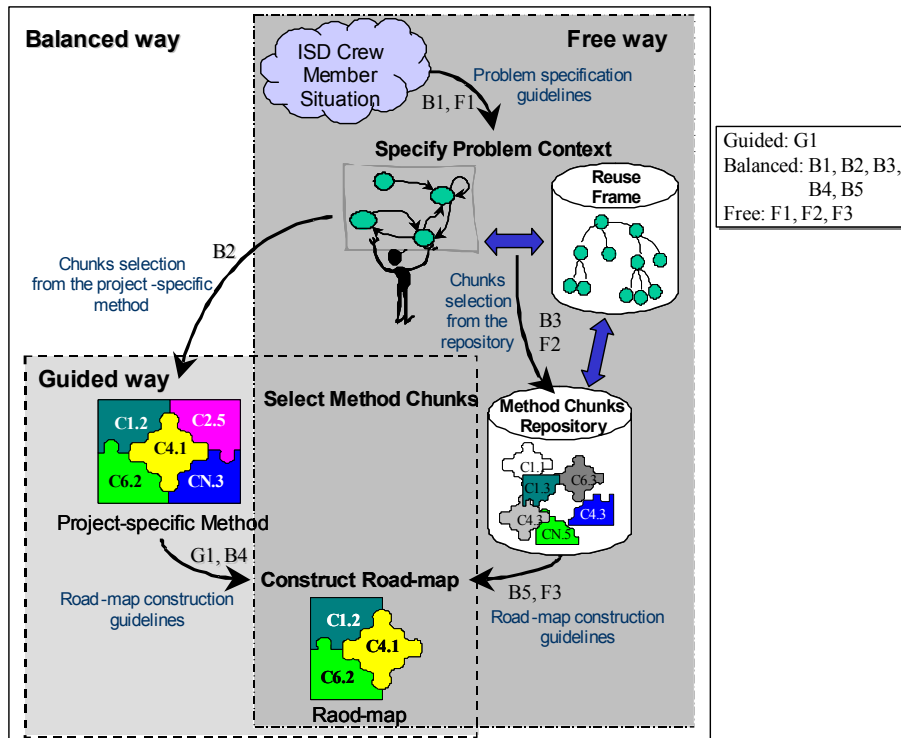


Fig. 12. Overview of the roadmap-driven method configuration approach

4.2.1 Guided Road-map Building

The guided way for the road-map building consists in selecting, in a manual way, method chunks from the project-specific method resulting from the first step of our approach. In other words, the ISD crew member selects different sections in the project-specific method map. In Fig. 12, this way is represented by the guideline G1.

4.2.2 Balanced Road-map Building

The balanced way for the road-map building consists first in specifying the ISD problem in terms of intention and context as it has been explained in section 4.1.1. Then, matching method chunks selected with the help of the situational metrics and following the tuning parameters given by the ISD crew member, as it has been explained in section 4.1.2. Then, alternative chunks and chunks matching the problem intention and problem context in the repository are also selected as it has been explained in section 4.1.3, including consistency check between method chunks from the project-specific method and method chunks from the repository. Tuning facilities may be used through these two last steps. Finally, the ISD crew member may proceed to an ultimate selection in a manual way, as suggested in the guided building way. In Fig. 12, the balanced way is represented by the set of guidelines B1, B2, B3, B4 and B5.

4.2.3 Free Road-map Building

This last way to build road-maps consists in selecting directly method chunks from the repository without the help of the project-specific method. Therefore, such a process starts by specifying the ISD problem in terms of intention and context as it has been explained in section 4.1.1. Then, the search is processed directly in the repository with the help of the situational metrics and following the tuning parameters given by the project team member, as it has been explained in section 4.1.2 and 4.1.4. Fig. 12 illustrates this way by the guidelines F1, F2 and F3.

5 Conclusion

In this paper we consider two perspectives of situational method engineering: (1) project-specific method engineering which aims to satisfy specific ISD project method requirements and (2) engineer-specific method engineering dedicated to satisfy requirements of an individual ISD crew member. We believe that these two perspectives are complementary and should be combined in order to support the ISD process at hand.

Therefore, in this work we combine two SME approaches, the assembly-based approach for project-specific method construction and the roadmap-driven approach for engineer-specific method configuration, as two core steps in the global SME approach. Even though, each of these approaches has been defined previously, independently one from the other, their integration is possible thanks to the concept of method chunk shared by both of them. A method chunks repository represents the basis of each of the two initial approaches. Therefore, the combination of the two approaches is possible if they share the same method chunks repository. We believe that only a minor unification effort is necessary to obtain the common method chunks repository, as the definitions of method chunks are similar in both approaches. Besides, the reuse frame is another common element in both of these two method engineering approaches.

The approach for assembly-based method construction, which represents the first step in our global SME approach, focuses on project-specific method building. This approach provides guidelines for: the specification of method requirements in line with the situation of the project at hand, the selection of the method chunks satisfying these requirements, and the assembly of the selected method chunks. The method requirements specification is based on the criteria defined in the reuse frame and allowing to characterise the situation of the project at hand. The reuse frame is also used during the method chunks selection process together with the product end process elements glossary in order to support the construction of queries. The assembly guidelines propose a set of assembly operators allowing to assemble product and process parts of the selected method chunks. Moreover, the similarity measures are provided in order to detect overlapping method chunks as well as to compare the method requirements with the solutions proposed by the selected chunks.

The approach for roadmap-driven method configuration represents the second step in our global SME approach and is dedicated to personalise the project-specific method defined previously for each ISD project crew member in order to support his/her specific tasks. Following this approach, the method chunks are selected from the project-specific method (defined during the first step) according to the requirements of a particular engineer and represent the corresponding engineers road-map. If it is necessary, additional method chunks can be retrieved from the method chunks repository and complete this road-map. Finally, in some specific cases, it could be necessary to select method chunks only from the repository without using project-specific method. Guidelines are provided to support the three ways for road-map construction called guided, balanced and free way respectively.

The assembly-based project-specific method building asks for a considerable knowledge about ISD methods, meta-modelling, method chunks assembly as well as it asks for a capability to evaluate the situation of the entire project and its method requirements. Therefore, this step should be realised by the method engineer or the manager of the current project. In the contrary, the approach for roadmap-driven method configuration can be executed by each ISD crew member.

It is evident, that the obtained global SME approach is more rich than the two initial ones as it allows to cover a large scale of situational method engineering requirements. To evaluate this

approach by applying it in real ISD projects is our current preoccupation. Two prototypes, one for each method engineering approach, have been developed. Our current objective is to develop software environment integrating these two approaches.

Besides, we are thinking about the method engineering ontology that should be richer than our reuse frame and glossary. This ontology is necessary to facilitate method chunks characterisation and formalisation, to permit their comparison, to position them in the method chunks repository, to define different relationships between method chunks, etc.

References

- [AHD00] The American Heritage Dictionary of the English Language. Houghton Mifflin Company 2000.
- [Brinkkemper98] Brinkkemper S., Saeki, M., Harmsen, F. (1998). Assembly Techniques for Method Engineering. *Proceedings of the 10th International Conference on Advanced Information Systems Engineering, CAiSE'98*, Pisa, Italy, LNCS 1413, Springer.
- [Cauvet01] Cauvet, C., Rosenthal-Sabroux, C. (2001). Ingénierie des systèmes d'information. Hermes.
- [Deneckère98] Deneckère, R., Souveyet, C. (1998). Patterns for extending an OO model with temporal features. *Proceedings of the International Conference on Object-Oriented Information Systems, OOIS'98*, Paris, France.
- [Firesmith02] Firesmith, D.G., Henderson-Sellers, B. (2002). The OPEN Process Framework – An Introduction. *Addison-Wesley*, Harlow, UK.
- [Graham97] Graham, I., Henderson-Sellers, B., Younessi, H. (1997). The OPEN Process Specification. *Addison-Wesley*, Harlow, UK.
- [Harmsen94] Harmsen A.F., Brinkkemper, S., Oei, H. (1994). Situational Method Engineering for Information System Projects. In *Olle T.W. and A.A. Verrijn Stuart (Eds.), Methods and Associated Tools for the Information Systems Life Cycle*, Proceedings of the IFIP WG8.1 Working Conference CRIS'94, North-Holland, Amsterdam, pp. 169-194.
- [Harmsen95] Harmsen A.F., Brinkkemper, S. (1995). Design and implementation of a method base management system for situational CASE environment. Proceedings of 2nd APSEC Conference, IEEE Computer Society Press, pp 430-438.
- [Harmsen97] Harmsen, A.F. (1997). *Situational Method Engineering*. Moret Ernst & Young.
- [Jacobson92] Jacobson I., M. Christenson, P. Jonsson, G. Oevergaard. (1992). Object Oriented Software Engineering: a Use Case Driven Approach. *Addison-Wesley*.
- [Jarke99] Jarke M., Rolland, C., Sutcliffe, A., Domges, R. (1999). The NATURE requirements Engineering. *Shaker Verlag*, Aachen.
- [Kumar92] Kumar, K., 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, pp257-268.
- [Mirbel02] Mirbel, I., de Rivieres, V. (2002). Adapting Analysis and Design to Software Context: The JECKO Approach. *Proceedings of the International Conference on Object-Oriented Information Systems, OOIS'02*, Montpellier, France, LNCS 2425, Springer, pp. 223-228..
- [Mirbel03a] Mirbel, I., de Rivieres, V. (2003). Conciliating User Interface and Business Domain Analysis and Design. *Proceedings of the 9th International Conference on Object-Oriented Information Systems, OOIS'03*, Geneva, Switzerland, LNCS 2817 Springer, pp. 383-399.
- [Mirbel03b] Mirbel, I., de Rivieres, V. (2003). Towards a UML profile for building on top of running software. In *UML and the Unified Process*. IRMA Press.
- [Mirbel04a] Mirbel, I. (2004). A polymorphic context frame to support scalability and evolvability of information system development processes. *Proceedings of the International Conference on Enterprise Information Systems, ICEIS'04*. Porto, Portugal.
- [Mirbel04b] Mirbel, I. (2004). Rethinking ISD Methods: Fitting project team members profiles. *I3S technical report I3S/RR-2004-13-FR*.
- [Plihon98] Plihon V., J. Ralyté, A. Benjamin, Maiden, N.A.M., Sutcliffe, A., Dubois, E. Heymans, P. (1998). A Reuse-Oriented Approach for the Construction of Scenario Based Methods. *Proceedings of the International Software Process Association's 5th International Conference on Software Process (ICSP'98)*, Chicago, Illinois, US.
- [Prakash99] Prakash, N. (1999). On Method Statics and Dynamics. *Information Systems*. Vol.34 (8), pp 613-637.
- [Prat97] Prat, N. (1997). Goal formalisation and classification for requirements engineering. *Proceedings of the 3rd International Workshop on Requirements Engineering: Foundations of Software Quality REFSQ'97*, Barcelona, Spain, pp. 145-156.

- [Punter96] Punter, H.T., Lemmen, K. (1996). The MEMA model: Towards a new approach for Method Engineering. *Information and Software Technology*, 38 (4), pp.295-305.
- [Ralyté01a] Ralyté J., Rolland, C. (2001). An Assembly Process Model for Method Engineering. *Proceedings of the 13th International Conference on Advanced Information Systems Engineering (CAISE'01)*, Interlaken, Switzerland, LNCS 2068, Springer, pp. 267-283.
- [Ralyté01b] Ralyté, J., Rolland, C. (2001). An approach for method reengineering. *Proceedings of the 20th International Conference on Conceptual Modeling (ER2001)*, Yokohama, Japan, LNCS 2224, Springer, pp.471-484.
- [Ralyté02] Ralyté, J. (2002). Requirements Definition for the Situational Method Engineering. *Proceedings of the IFIP WG8.1 Working Conference on Engineering Information Systems in the Internet Context (EISIC'02)*, Kanazawa, Japan. Kluwer Academic Publishers, pp.127-152.
- [Ralyté04] Ralyté, J., Rolland, C., Deneckère, R. (2004). Towards a Meta-tool for Change-centric Method Engineering: a Typology of Generic Operators. *Proceedings of the 16th International Conference CAISE'04*, Riga, Latvia. LNCS 3084, Springer, pp. 202-218.
- [Ralyté99a] Ralyté, J., Rolland, C., Plihon, V. (1999). Method Enhancement with Scenario Based Techniques. *Proceedings of the 11th International Conference on Advanced Information System Engineering (CAISE'99)*, Heidelberg, Germany. LNCS 1626, Springer, pp. 103-118.
- [Ralyté99b] Ralyté, J. (1999). Reusing Scenario Based Approaches in Requirement Engineering Methods: CREWS Method Base. *Proceedings of the 10th International Workshop on Database and Expert Systems Applications (DEXA'99)*, 1st International Workshop on the Requirements Engineering Process – Innovative Techniques, Models, Tools to support the RE Process (REP'99). Florence, Italy, IEEE Computer Society, pp. 305-309.
- [Rolland00] Rolland, C., Nurcan, S., Grosz, G. (2000). A Decision Making Pattern for Guiding the Enterprise Knowledge Development Process. *Journal of Information and Software Technology*, 42, pp. 313-331.
- [Rolland96a] Rolland, C., Plihon, V. (1996). Using generic chunks to generate process models fragments. *Proceedings of 2nd IEEE International Conference on Requirements Engineering, ICRE'96*, Colorado Spring.
- [Rolland96b] Rolland, C., Prakash, N. (1996). A proposal for context-specific method engineering. *Proceedings of the IFIP WG 8.1 Conference on Method Engineering*, Chapman and Hall, Atlanta, Gerorgie, USA, pp 191-208.
- [Rolland98] Rolland, C., Plihon, V., Ralyté, J. (1998). Specifying the Reuse Context of Scenario Method Chunks. *Proceedings of the 10th International Conference on Advanced Information System Engineering (CAISE'98)*, Pisa, Italy. LNCS 1413, Springer, pp. 191-218.
- [Rolland98a] Rolland C., Souveyet, C., Ben Achour, C. (1998). Guiding Goal Modelling Using Scenarios. *IEEE Transactions on Software Engineering*, 24 (12), pp. 1055-1071.
- [Rolland98b] Rolland C., Ben Achour, C. (1998). Guiding the construction of textual use case specifications. *Data & Knowledge Engineering Journal*, 25(1), pp. 125-160.
- [Rolland99] Rolland, C., Prakash, N., Benjamen, A. (1999). A Multi-Model View of Process Modelling. *Requirements Engineering Journal*, Vol. 4 (4), pp169-187.
- [Saeki93] Saeki, M., Iguchi, K., Wen-yin, K., Shinohara, M. (1993). A metamodel for representing software specification & design methods. *Proceedings of the IFIP WG8.1 Conference on Information Systems Development Process*, Come, pp 149-166.
- [Slooten93] van Slooten K., Brinkkemper, S. (1993). A Method Engineering Approach to Information Systems Development. In *Information Systems Development process*, N. Prakash, C. Rolland, B. Pernici (Eds.), Elsevier Science Publishers B.V. (North-Holand), pp. 167-186.
- [Slooten96] van Slooten, K., Hodes, B. (1996). Characterizing IS Development Projects. *Proceedings of the IFIP WG 8.1 Conference on Method Engineering*, Chapman and Hall, pp. 29-44.
- [Tawbi98] Tawbi, M., Souveyet, C., Rolland, C. (1998). L'ECRITOIRE a tool to support a goal-scenario based approach to requirements engineering, *Information and Software Technology journal*. Elsevier Science B.V.
- [UML] Object Management Group, *Unified Modelling Language (UML)*, version 1.5. Available at <http://www.omg.org/technology/documents/formal/uml.htm> 2004.

Annexe: Reuse Frame

In this annexe we present our reuse frame. The top level of our reuse frame is presented in Fig. 9. Fig. 13 shows the branch *application domain* of the reuse frame. Fig. 14 defines the family of *contingency factors*, the branch *system engineering activities* is presented in Fig. 15 and finally Fig. 16 shows the aspects associated to the *project management* family.

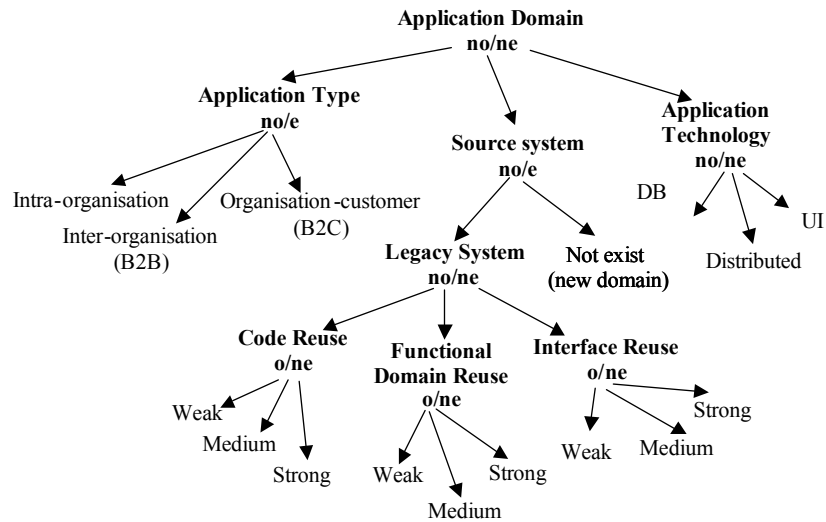


Fig. 13. Reuse Frame: Application Domain

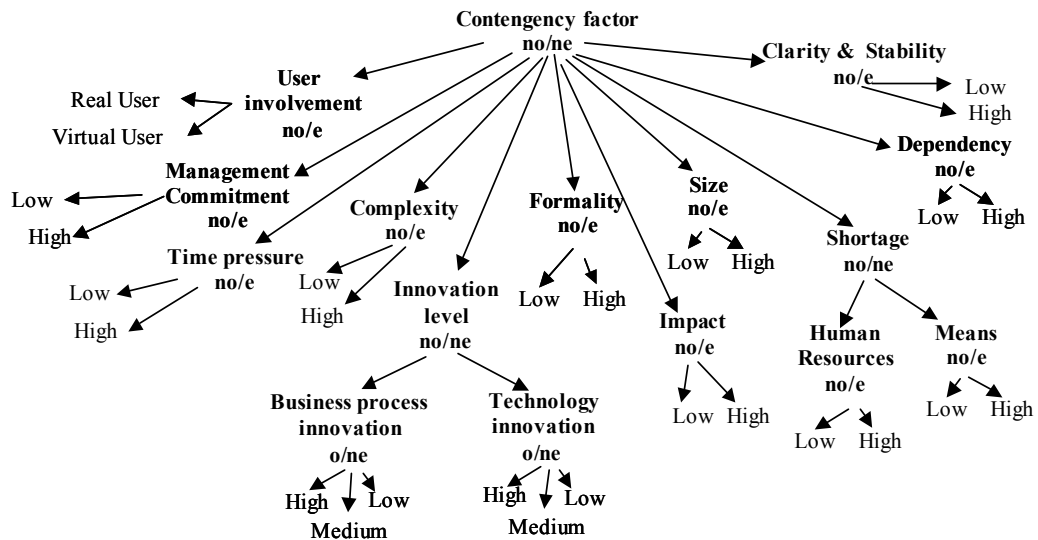


Fig. 14. Reuse Frame: Contingency factors

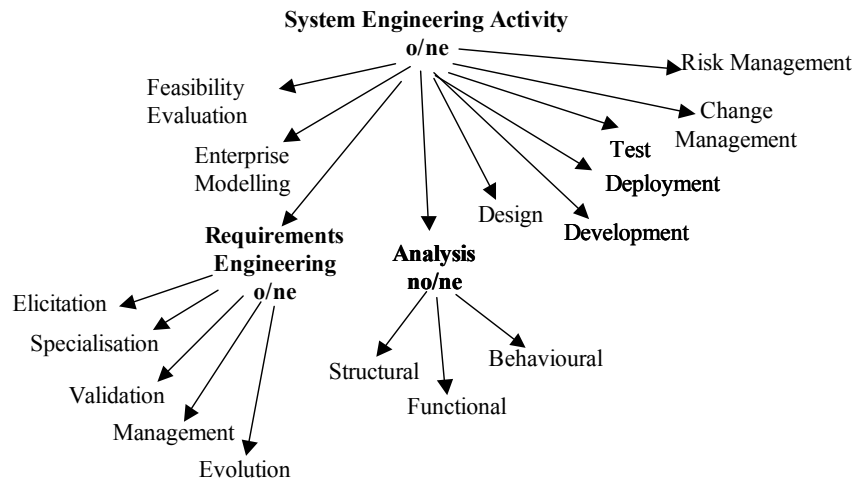


Fig. 15. Reuse Frame: System Engineering Activities

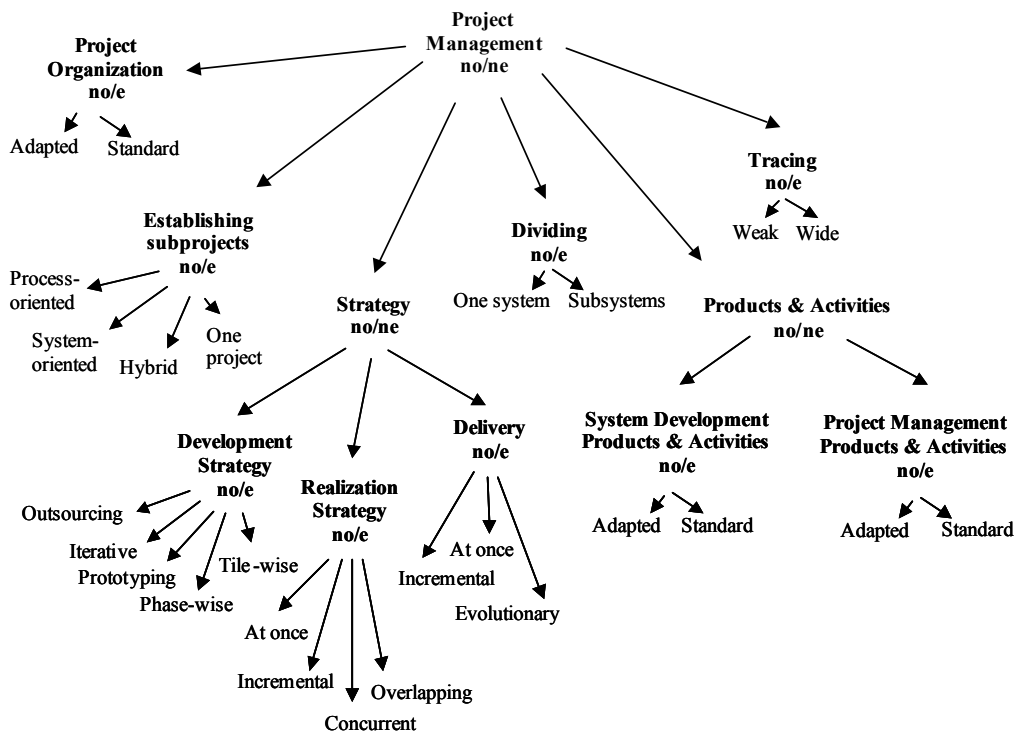


Fig. 16. Reuse Frame: Project Management