

LABORATOIRE



INFORMATIQUE, SIGNAUX ET SYSTÈMES
DE SOPHIA ANTIPOLIS
UMR 6070

RETHINKING ISD METHODS: FITTING PROJECT TEAM MEMBERS PROFILES

Isabelle MIRBEL

Projet EXECO

Rapport de recherche
ISRN I3S/RR-2004-13-FR

Avril 2004

RÉSUMÉ :

MOTS CLÉS :

Système d'information, Ingénierie des méthodes, méthode situationnelle, réutilisation, fragment

ABSTRACT:

The increasing complexity of Information Systems (IS) asks for rethinking IS development (ISD) methods and their usage. Dedicated efforts have already been made to decompose ISD methods into process fragments and to propose to method engineers techniques and operators to allow them to adapt existing methods or to build new ones. But customization efforts have also to be provided to project team members who are using methods while performing their daily tasks. Our research work deals with an approach about method customization 'on the fly' to match as well as possible the profiles of project team member tasks within the project. Our purpose is not to propose a new way to build methods, as several approaches already exist on this topic, but to ease the use of existing ones by making them less rigid, allowing their adaptation to the need of the company, the project and most of all, the project team member.

Our framework is made of a repository of reusable process fragments to capitalize and share guidelines about ISD. Guidelines are entered by project team members accumulating knowledge about their own experiences in ISD or by method engineers breaking down well-known methods into process fragments.

In order to fully exploit the potential of fragmentation and customization, we believe knowledge about organizational, technical and human factors, which are critical aspects of ISD, have to be taken into consideration. Therefore, we introduce the Reuse Frame, a scalable and polymorphic ontological structure, which allows project team members, when they enter guidelines as process fragments into the repository, to specify each of them with regards to the aspects captured in the Reuse Frame through the notion of process fragment reuse context. Then, thanks to the notion of problem reuse situation, project team members better qualify their ISD problem in order to find a suitable solution to their methodological need. A solution is made of process fragments, which fragment reuse contexts match the problem reuse situation, organized into a road-map specially built to answer the project team member need and directly usable by him/her.

In this paper, we start first by discussing the Reuse Frame and the process fragmentation. Then we show how method customization is handle through process fragment selection, tuning and road-map building.

KEY WORDS :

Information System, Method Engineering, Situational Method, Reuse, Fragment

Rethinking ISD Methods: Fitting project team members profiles

Isabelle Mirbel

Laboratoire I3S

Route des Lucioles - BP 121
06903 Sophia Antipolis Cedex

France

Phone: (+33) 4 9294 2760

Fax : (+33) 4 9294 2896

Isabelle.Mirbel@unice.fr

Abstract

The increasing complexity of Information Systems (IS) asks for rethinking IS development (ISD) methods and their usage. Dedicated efforts have already been made to decompose ISD methods into process fragments and to propose to method engineers techniques and operators to allow them to adapt existing methods or to build new ones. But customization efforts have also to be provided to project team members who are using methods while performing their daily tasks. Our research work deals with an approach about method customization 'on the fly' to match as well as possible the profiles of project team member tasks within the project. Our purpose is not to propose a new way to build methods, as several approaches already exist on this topic, but to ease the use of existing ones by making them less rigid, allowing their adaptation to the need of the company, the project and most of all, the project team member.

Our framework is made of a repository of reusable *process fragments* to capitalize and share guidelines about ISD. Guidelines are entered by project team members accumulating knowledge about their own experiences in ISD or by method engineers breaking down well-known methods into *process fragments*.

In order to fully exploit the potential of fragmentation and customization, we believe knowledge about organizational, technical and human factors, which are critical aspects of ISD, have to be taken into consideration. Therefore, we introduce the *Reuse Frame*, a scalable and polymorphic ontological structure, which allows project team members, when they enter guidelines as *process fragments* into the repository, to specify each of them with regards to the aspects captured in the *Reuse Frame* through the notion of *process fragment reuse context*. Then, thanks to the notion of *problem reuse situation*, project team members better qualify their ISD problem in order to find a suitable solution to their methodological need. A solution is made of *process fragments*, which *fragment reuse contexts* match the *problem reuse situation*, organized into a *road-map* specially built to answer the project team member need and directly usable by him/her.

In this paper, we start first by discussing the *Reuse Frame* and the process fragmentation. Then we show how method customization is handle through process fragment selection, tuning and road-map building.

Rethinking ISD Methods: Fitting project team members profiles

1 Introduction

Information Systems Development (ISD) evolves continually, creating new challenges especially in terms of methods. Looking at the way Information System Development (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 method tailoring. One way to support this customization is the assembly of predefined process fragments. Dedicated efforts have been made, in the field of method engineering, to decompose ISD methods into process fragments [1, 8].

Indeed customization of ISD methods have mainly be thought of for the person in charge of building a new method, i.e. the method engineer, in order to allow him/her to adapt the method to the need of its company or ISD project(s). But there is also a need for customizations dedicated to each person using the method, i.e. project team member, to provide each of them with dedicated guidelines (or heuristics) which are to be followed while performing their daily tasks.

The approach presented in this paper aims at customizing a method 'on the fly' in order to match as well as possible the profile of the project team member job within the project [14, 3, 7]. For this purpose, efficient classification and retrieving means to store and select process fragments have to be provided. Efforts have already been made on this topic in the fields of method engineering. These classification and retrieving techniques are currently based on structural relationships among process fragments (specialization, composition, alternative, ...) and keyword matching [2, 9]. From our point of view, current classification and retrieving means do not fully exploit the potential of breaking down ISD methods into process fragments and tailoring them, especially when populating the process fragment repository as well as when retrieving them. We believe knowledge about organizational, technical and human factors, which are critical knowledge about ISD [2], should be taken into consideration in addition to structural knowledge and keywords. It allows to better qualify process fragments when entering them into the repository and to enable the use of more powerful matching techniques to find them again when looking for ISD methodological support. It also allows to better express methodological needs for a specific ISD project, improving this way the chance to get adequate and useful process fragments. 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). Figure 1 summarizes our approach.

In this paper, we start first by discussing the critical aspects of IS and we show how to handle them through method customization. Then, the repository of predefined process fragments is described in section 3. The method customization process is presented in section 4. Finally, we conclude in section 5.

2 Handling critical aspects of information systems

To better take advantage of ISD method customization, we believe IS critical aspects have to be taken into consideration. Process fragments have to be defined with regards to IS critical aspects in order to be more easily reused in similar methodological situations. Process fragments, as well as methodological needs, may be defined more or less precisely with regards to ISD critical aspects.

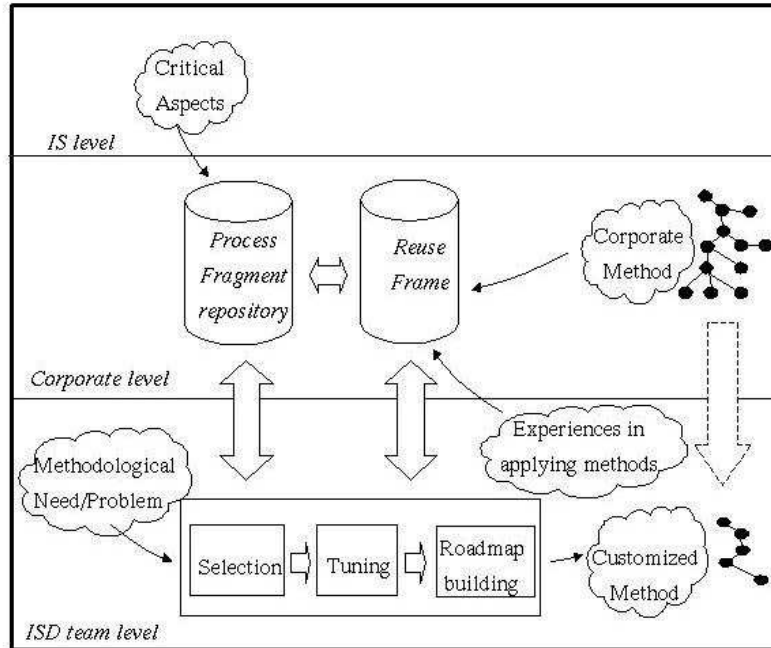


Figure 1: Method customization

Therefore, the three main aspects: human, organizational and technical, are refined and presented in a way supporting polymorphism through a tree of successively refined aspects. By providing such an ontological structure, we enlarge the panel of means for method engineers to drive project team members on the way to apply a method and to broadcast ways of working which is most of the time reduced to deliverables. Our approach provides to project team members means to find the most suitable information with regards to his/her ISD methodological need (or problem).

2.1 The Reuse Frame

ISD critical aspects are described in terms of *aspects*, belonging to *aspect families*, which are successive refinements of the three main factors of IS: human, organizational and technical.

An *aspect* is a leaf node in the tree. A *family* of aspect is a non-leaf node with at least two sub-nodes (which could be *aspect* or *family* nodes). Families are interesting to allow a better understanding of aspects entered in the framework, because they provide a way to group and organize them. With regards to the *organizational* dimension, we started from the work of K. van Slooten and B. Hodes providing elements to characterize ISD projects [13]. 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 the software development process [7]. And finally, about the *human* dimension, we currently propose a basic description of the expertise of the project team members (low, medium or high). ISD critical aspects are reassembled into a tree called the *Reuse Frame*, where leaf-nodes are aspects and intermediary nodes are families. As an example, the technical branch of the *Reuse Frame* is presented in Figure 2. In this branch, we distinguish the following features: the application to be developed includes a user-interface (UI), a database (BD), is distributed (**Distributed**) or is built on top of a running application (**Running software**). This last aspects is presented as a *family* because different aspects of a running software may be considered : **Functional domain**, **Interface** and **Code** [7]. Again a distinction is done among

weak, medium and strong reuse of existing code, functional domain and interface.

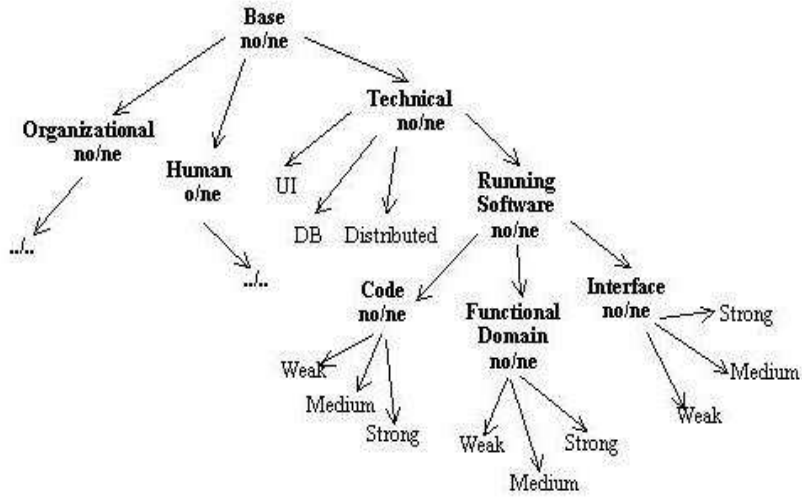


Figure 2: Technical branch of the *Reuse Frame*

Indeed, nodes close to the root node are seen as general aspects while node close to leaf nodes (including leaf-node) are seen as precise aspects.

A leaf-node is defined through a *name*. An intermediary node is also defined through a *name* and completed by information about relationships among the different aspects or sub-families belonging to it: the fact that the different aspects may be *exclusive* (or not) and/or *ordered* (or not). For the full definition of *Reuse Frame* elements, refer to [6]. The **Software to develop** aspect presented in Figure 2, for instance, is described as *non-ordered* because the different sub-families are non-related aspects. They are described as *non-exclusive* because they may be associated with a same problem, solution or fragment. **Weak**, **Medium** and **Strong** aspects are considered as *non-ordered* because a fragment dedicated to guidelines to weakly keep the code for instance may not necessary be complementary to the one dedicated to keep the code in a medium way. And on the contrary, it is not forbidden also to search for a solution including for instance **Medium** and **Strong** aspects. Therefore, they are specified as *non-exclusive*.

Evolution is a key issue of IS [5] and therefore also a key issue of tools and techniques supporting their development and maintenance. Method engineers could make the *Reuse Frame* evolves when they specify more deeply existing aspects by refining them (i.e. changing aspects into families grouping new aspects) or when they reorganize existing families by adding new intermediary nodes. Of course, in this case, fragments associated to refined aspects may have to be associated again to new nodes by method engineers.

The *Reuse Frame* allows method engineers to drive the project team 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 Taking advantage of the *Reuse Frame*

The ISD critical aspects defined in the *Reuse Frame* aim at improving matching techniques when retrieving process fragments to answer a specific need. Therefore, means have to be provided to

select from the *Reuse Frame* the pertinent aspects to be associated to process fragments and to methodological problem (and its solutions) in order to enable powerful matching techniques. It is done respectively through the notion of *Process Fragment Reuse Context* and *Problem Reuse Situation*.

Process Fragment Reuse Context and **Problem Reuse Situation** are defined with the help of *criteria*.

A **criterion** is defined as a path from the root node **base** to a node n_n of the *Reuse Frame* (if n_n is a *family* node, then its *exclusion* field is different from e).

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 .

The **Process Fragment Reuse Context** is indeed a set of *criteria*.

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

Process fragments providing general guidelines are usually characterized by 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 process fragments described through precise criteria, that is to say paths ended by nodes from the *Reuse Frame* close to leaf nodes or leaf nodes themselves.

In the *Problem Reuse Situation*, in addition to the pertinent criteria, called *necessary criteria*, a project team member may need to give *forbidden criteria*, that is to say aspects he/she is not interested in. It could be helpful in some cases to be sure the process fragments including these (forbidden) aspects will not appear in the solution answering the methodological need.

A **Problem Reuse Situation** is defined as a set of at least one compatible *necessary criteria* and a set of compatible *forbidden criteria*. All criteria are taken from the *Reuse Frame* and there is no common criterion between the two sets.

3 Breaking down methods into fragments

When developing IS, heuristics are elaborated and may be useful to other teams facing close situations in different projects independently of the functional domain as well as the technical domain. It could be useful for ISD project team members to be able to retrieve heuristics accumulated by other ISD project team member to take advantage of the way they found to succeed in solving problems and to help them to focus on critical aspects of ISD, especially with regards to ways of working. Therefore, approaches have been developed to specify methods [4, 11] and to break them down into fragments [14]. Efforts have also been made to formalize fragment definition (or pattern definition) [10, 12]. A classification of the different approaches may be found in [9].

In addition to the different specification levels that are provided in the literature about process fragments, different objectives are targeted by the approaches. A first family of approaches aims at documenting methods through well-defined fragments [12]. These approaches do not provide powerful supports nor to reuse the fragments from one method to another nor to customize the

method for a specific project or group of persons. Their strength resides in the effort of specification with regards to the elements a method is made of (tasks, activities, resources, etc.). A second family of approaches groups works which aim is to help in building new methods starting from existing ones (instead of building them from scratch) [8, 1]. In this kind of approaches, the focus is on the operators (add, remove, unify or merge) provided to allow a new combination of existing process fragments and on mechanisms to evaluate the similitude among them. Both combination operators and evaluation mechanisms are dedicated to method engineers. But, project team members also need to benefit, through reuse and adaptation mechanisms, from the experiences acquired during the resolution of previous problems in terms of applying the method in a concrete way (i.e. using it). Therefore, a third family of proposals, our work is included in, focuses on method fragmentation for project team members, to provide them with guidelines (or heuristics) which are to be reused while performing their daily task [3, 7].

The process fragments corresponding to a methodological need are presented in a sequential way constituting the road-map to be followed while applying a method (i.e. applying successively a set of process fragments). In the following, we discuss process fragments and road-maps.

3.1 Process fragments

Process fragments are the result of (i) the capitalization of their own experience in ISD directly by project team members, or (ii) the breaking down of known methods by method engineers [8].

A **Process Fragment** f_n is a 6-uple, $f_n = \langle nf_n, RC_n, i_n, g_n, Ass_n, Inc_n \rangle$, where

- nf_n is the **name** of the fragment. This name should be meaningful to indicate its purpose. Fragment name is unique in order to identify the fragment.
- RC_n is the **Process Fragment Reuse Context** associated to the fragment as previously defined (cf section 2.2).
- i_n is the **intention** of the fragment, that is, a textual description of the goal of the fragment and a list of keywords. It is defined as a 2-uple $\langle des_{i_n}, KW_{i_n} \rangle$ where:
 - des_{i_n} is the textual description
 - KW_{i_n} is the set of keywords associated to the fragment.
- g_n is the **guidelines** of the fragment. It is defined as "a statement or other indication of policy or procedure by which to determine a course of action" [8]. For us, a guideline is a set of pair $\langle n, h \rangle$ where:
 - n is the notation used to express guidelines
 - h is the textual description of the guidelines
- Ass_n is the set of fragments **associated** with the current fragment. It is defined as a set of 3-uple $\langle fr-ass, re-ass, deg-ass \rangle$ where
 - $fr-ass$ is a fragment sharing at least part of its *Process Fragment Reuse Context* with the current fragment f_n
 - $re-ass$ qualifies the relationship between f_n and $fr-ass$. We distinguish 2 kinds of relationships: *complementarity* and *alternative*.

- *deg-ass* quantifies the relationship between f_n and *fr-ass*. $deg-ass \in [0, 1]$. *deg-ass* is a *necessity* degree when associated to the *complementarity* relationship. It is a *clothesness* degree when associated to the *alternative* relationship.

The *complementarity* relationship is bijective: If a fragment f_1 appears as complementary to f_2 , then f_2 also appears as a complementary fragment of f_1 with the same degree.

The *alternative* relationship is not bijective since a fragment f_1 may include another fragment f_2 and therefore be an alternative to it. But the contrary would be wrong.

Close intentions are required if *deg-ass* is less than 1, identical ones are requested otherwise.

- Inc_n is the set of **not compatible fragments** associated with the current fragment. It is defined as a set of pair $\langle fr-inc, deg-inc \rangle$ where
 - *fr-inc* is a fragment not compatible with the current fragment f_n
 - *deg-inc* is the degree of *incompatibility* between f_n and *fr-inc*.

An example of process fragment is given in figure 3. In this figure, a process fragment named **Requirement-out-of-scope** is presented. Its *Process Fragment Reuse Context* indicates it shows guidelines for when developping on top of a running software. The process fragment has an associated fragments **DB-out-of-scope**, which is complementary. It does not have incompatible fragments. The intention explains the purpose of the process fragment which is to help in documenting the running part of the application under which the development will take place. Associated guidelines in UML are then given.

| | |
|---|---|
| Name | Requirement-Out-of-Scope |
| Reuse Context | {[base – software – running software]} |
| Related Fragments | BusinessDomain-Out-of-Scope – complementarity – 0.75 |
| Non-compatible Fragments | – |
| Intention To document existing parts of the running software useful to understand the purpose of the new software but not directly related to the new development. | |
| Guidelines | Notation UML use–case diagrams |
| | Description Add use–cases dedicated to running functionalities increasing the understanding of the software. Stereotype them with <<out-of-scope>>. Group all the use–cases stereotyped <<out-of-scope>> in a package (or set of packages) also stereotyped <<out-of-scope>> |

Figure 3: An example of process fragment

3.2 Road-maps

Different ways may be provided, even inside a same method, to satisfy an engineering goal. Moreover, the sequence through which process fragments have to be used is not always pre-determined: they may or not be related by a precedence relationships. Therefore, different road-maps among a set of process fragments are possible. A road-map is composed of one or several coherent sequence(s) of process fragments corresponding to the usage of the method by a particular project team member. It is a customized view of the method to answer the specific need of project team

member. Our approach about method configuration aims at finding the road-map which is the most suitable for the specific need of a project team member.

To be added to the repository, a new process fragment must be linked to process fragments already stored in the repository. Two process fragments are linked when they can be used inside a same method. The *precedence metric* allows to estimate the sequence in which the process fragments have to be applied.

Let f_a and f_b be 2 process fragments. The *precedence metric* is composed of 3 values :

- pb is the probability for f_a before f_b
- pa is the probability for f_a after f_b
- pi is the probability for f_a and f_b to be in an indefinite sequence

with $pb, pa, pi \in [0, 1]$ and $\sum pb, pa, pi = 1$.

For each process fragment f_b linked to f_a , the project team member introducing f_a gives the *precedence metric* values. Possible values are:

- $\{pb=0, pa=0, pi=1\}$: the sequence between f_a and f_b is not significant;
- $\{pb=1, pa=0, pi=0\}$: the project team member indicates f_a should be useful before f_b ;
- $\{pb=0, pa=1, pi=0\}$: symmetrically, f_b should be useful before f_a .

Figure 4 presents an example of road-map taken from [7] dedicated to the requirement phase of ISD customized to build on top of a legacy application. In this figure, fragments with an X are dedicated to guidelines to deal with ISD on top of a legacy application while bold-line fragments provide general guidelines.

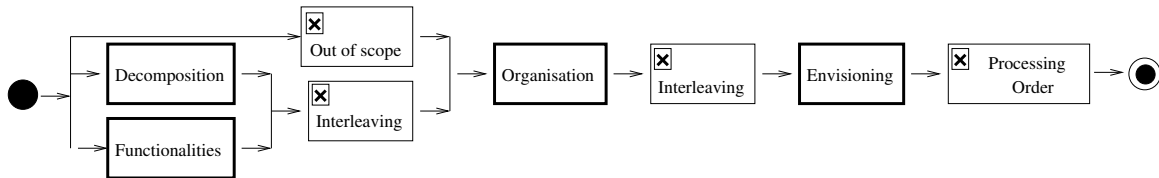


Figure 4: Example of road-map

In our approach, process fragments stored in a dedicated repository and organized into road-map allow capitalization of knowledge about the resolution of methodological problems. Indeed, the *Reuse Frame* provides knowledge *for* reuse, while reusable knowledge is stored in the repository of predefined process fragments. In the next section, we show how to take advantage of this framework through method customization.

4 Method customization

In our approach for method customization, the project team member looking for a dedicated configuration of the ISD project method starts by expressing its specific problem, mainly with the

help of the *Reuse Frame*. Then, the process fragments matching the problem are retrieved from the repository of predefined process fragments. Tuning facilities are provided to the project team member to let him/her get the most suitable road-map within the method to answer its specific need. And finally, the selected process fragments are ordered to compose the road-map, that is to say the guidelines to be successively applied, in order to take as much advantage as possible of the method.

4.1 Selecting the right process fragments

First of all, the project team member expresses its problem in terms of applying the method. The problem is specified through a *Problem Reuse Situation* (cf section 2.2) and eventually a set of keywords that the selected process fragments should match.

Thanks to the *Process fragment Reuse Context* associated to each process fragment stored in the repository and the *Problem Reuse Situation* given by the project team member, matching techniques could be applied to select process fragments corresponding to the problem. Therefore, we introduce the *situational metric* allowing to quantify the matching between a process fragment and a problem. This metric is based on (i) the number of common criteria between the necessary criteria from the *Problem Reuse Situation* and the *Process fragment Reuse Context*, (ii) the number of common criteria between the forbidden criteria from the *Problem Reuse Situation* and the *Process fragment Reuse Context*, (iii) the number of required necessary criteria from the *Problem Reuse Situation*:

$$ms(pb, pf) = \frac{card(RC_{pf} \cap CN_{pb}) - card(RC_{pf} \cap CF_{pb})}{card(CN_{pb})}$$

where pb is a problem, CN_{pb} the necessary criteria from its *Problem Reuse Situation*, CF_{pb} the forbidden criteria from its *Problem Reuse Situation*; pf is a process fragment, RC_{pf} its *Process Fragment Reuse Context*.

A positive value of $ms(pb, pf)$ indicates that there are more necessary criteria than forbidden ones in the process fragment pf under consideration with regards to the problem pb . On the contrary, a negative value of $ms(pb, pf)$ indicates that there are less necessary criteria than forbidden ones. The perfect adequation is represented by the value 1 and the worst situation by the value $\frac{-card(RC_{pf} \cap CF_{pb})}{card(CN_{pb})}$.

4.2 Tuning the selection

The *Reuse Frame* introduced previously (cf section 2) supports different levels of granularity with regards to criterion definition and use. Criteria which ending node is close to the *base* node are much more generic than criteria which ending node is close to leaf nodes. Indeed, process fragments providing general guidelines are usually associated to general criteria, while specific guidelines are provided in process fragments associated to precise criteria.

4.2.1 Tuning necessary criteria

When searching for predefined process fragments into the repository, a project team member is interested in process fragments which criteria strictly match the necessary criteria from the *Problem Reuse Context*. But process fragments including more specific criteria in their *Reuse Contexts* may also be interesting: process fragments associated to more specific criteria usually provide more specific guidelines. They may better cover part of the methodological problem the project team member has to deal with.

if one looks for instance at process fragments matching the [base - technical - running software - functional domain] criterion, he/she may be also interested by the process fragments matching the [base - technical - running software - functional domain - weak], [base - technical - running software - functional domain - medium] and [base - technical - running software - functional domain - strong] criteria (cf Figure 2).

In the same way, the project team member may be interested in process fragments associated to more general criteria usually providing more general-purpose guidelines which could also be useful. If one still looks for instance at process fragments matching the [base - technical - running software - functional domain] criterion, he/she may also be interested by the process fragments matching the [base - technical - running software] and the [base - technical] criteria (cf Figure 2).

4.2.2 Tuning forbidden criteria

Taking into consideration process fragments associated to 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*; and enlarging the set of forbidden criteria to more specific criteria means to forbid process fragments associated to too specific criteria, usually present in reuse contexts of process fragments providing too specific guidelines.

4.2.3 Synthesis

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 provides a way for the project team member to reduce or enlarge the number of process fragments of the solution to its problem. If one feel he/she did not find enough process fragments with regards to its methodological need, he/she may allow more general and/or more specific criteria in order to find more process fragments. On the contrary, if the set of process fragments provided as an answer to its 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 process fragments associated to these criteria.

The following table summarizes the tuning possibilities.

| | Strict selection (default) | Extended selection | |
|-----------------------|------------------------------------|--|---|
| | | with general criteria | with specific criteria |
| Necessary criteria | To search for process fragments | To retrieve more process fragments | |
| | | General process fragments | Specific process fragments |
| Forbidden criteria | To avoid process fragments | To forbid more process fragments | |
| | | Avoid full branches of the <i>Reuse Frame</i> | Avoid too specific process fragments |

When computing the *situational metric* between a process fragment pf and a problem pb , criteria included as more general or more specific ones have to be weighted with regards to their distance from the criteria given in the problem definition. Therefore, the *situational metric* becomes:

$$ms(pb, pf) = \frac{\sum_{card(CNE)}^{i=1} P_{c_i} - \sum_{card(CFE)}^{j=1} P_{c_j}}{\sum_{card(CN)}^{k=1} P_{c_k}}$$

with CN the necessary criteria, CNE the extended necessary criteria also present in RC_{pf} , CF the forbidden criteria and CFE the extended forbidden criteria of the problem pb also present in RC_{pf} ; and with P the criteria weight computed as follows:

$$\text{if } c_i \in CN, p_{c_i} = 1 \text{ else } p_{c_i} = \frac{1}{2^{nbn}}$$

where nbn is the number of node between cr and c_i excluding cr and including c_i .

Finally, a set of process fragments is obtained as result of the search.

4.3 Building the road-map

From the selection and tuning steps of our customization process, we get a set of process fragments. Indeed, these process fragments may not be related all together (through the precedence relationship or as associated process fragments). They may for instance cover disjoint stages of the method (at the beginning and the end of the method for instance). So, the result set of process fragments is indeed made of different subsets. In each subset are reassembled process fragments related to each others as *associated* process fragments or through the precedence relationship.

4.3.1 Adding complementary process fragments

In the repository, process fragments may be linked to each other as *associated* process fragments (cf section 3.1. Degrees are also associated to this kind of association to quantify the complementarity. When building the road-map corresponding to each subset of the result set, the coherency of the selected process fragments has to be enforced by fulfilling the constraints expressed through this kind of relationship: complementary process fragments which degrees are higher than a predefined threshold have to be added to the solution. Indeed, the project team member tunes the level of coherency he/she wants its roadmap to satisfy by setting the *complementarity* threshold. A high complementarity threshold leads to a solution in which only very complementary process fragments are added, while a low one leads to a solution in which most of the complementary process fragments are included.

4.3.2 Removing not compatible fragments

In the same way, process fragments may also be linked to each other as *not compatible* process fragments. In this case, degrees quantify the incompatibility. When building the road-map corresponding to each subset of the result set, the coherency of the selected process fragments has to be enforced by fulfilling the constraints expressed through these incompatibility relationships: not compatible process fragments which degrees are higher than a predefined threshold have to be removed from the solution. Indeed, the project team member tunes the level of coherency he/she wants its roadmap to satisfy by setting the *incompatibility* threshold. A high incompatibility threshold leads to the removal of very incompatible process fragments from the solution, while a low threshold leads to the removal of most of the incompatible process fragments.

Tuning the road-map building process provides again a way for the project team member to reduce or enlarge the number of process fragments included in the solution to his/her problem.

4.3.3 Removing still inconsistent process fragments

During the selection step, *Process Fragment Reuse Contexts* are evaluated with the help of the *situational metric* to be or not kept as part of the result set. Complementary process fragments added to the solution for coherency purposes have to be evaluated with the *situational metric*.

Only complementary process fragments which *situational metric* is positive are kept. And if not kept, the process fragments requiring them as complementary process fragment remain still incoherent and have therefore to be removed from the solution.

Finally, the set of process fragments still kept in each subset constitutes the road-map(s) corresponding to the project team member problem and is presented to him/her as a succession of guidelines to be successively applied. The full selection process is summarized in Figure 5.

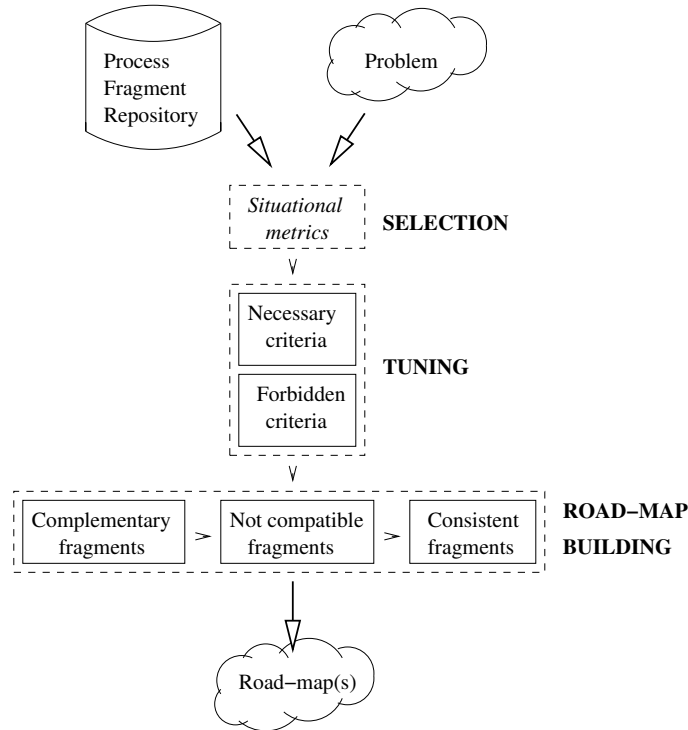


Figure 5: Selection process

5 Conclusion

In this paper we presented an approach to customize method in order to fit the need of the project team member. Customization is supported by method fragmentation into reusable process fragments. In addition, we proposed the *Reuse Frame*, a framework grouping aspects representing critical aspects of IS and allowing method engineers to specify process fragments meaningful features and project team member to select the right process fragments. After discussing the usefulness of the *Reuse Frame* and the way to break down methods into fragments we focused on the customization process and its main 3 steps: selection, tuning and road-map building, allowing project team member to get a customized point of view on the IS project method in order to fit his/her own methodological need.

Attempts have already been made to use such an approach on real projects, especially with regards to the technical dimension of IS [7]. And a case tool is under development to validate the whole approach.

In the future, our efforts will focus on the exploitation of tracking information about the way project team member reuse the process fragments and the feedback they may give on their reuse experience: some process fragments, for instance, may be more useful than others, fragments may indeed be skipped or removed from road-maps. The *precedence metric* should be updated in consequence to also reflect the experience capitalized through the reuse of the process fragments. In the same way, we would like to weight process fragments with regards to the expertise level of the project team members introducing the process fragments into the repository.

References

- [1] S. Brinkkemper, M. Saeki, and F. Harmsen. Assembly techniques for method engineering. In *10th International Conference on Advanced Information Systems Engineering*, Pisa, Italy, 1998.
- [2] C. Cauvet and C. Rosenthal-Sabroux. *Ingenierie des systemes d'information*. Hermes, 2001.
- [3] M. Gnatz, F. Marschall, G. Popp, A. Rausch, and W. Schwerin. Modular process patterns supporting an evolutionary software development process. *Lecture Notes in Computer Science*, 2188, 2001.
- [4] Object Management Group. Software process engineering metamodel. <http://www.omg.org/>.
- [5] D. Konstantas, M. Leonard, Y. Pigneur, and S. Patel, editors. *Object-Oriented Information Systems*. Springer-verlag, 2003.
- [6] I. Mirbel. A polymorphic context frame to support scalability and evolvability of information system development processes. In *6th International Conference on Enterprise Information Systems*, april 2004.
- [7] I. Mirbel and V. de Rivieres. Adapting Analysis and Design to Software Context: the JECKO Approach. In *8th International Conference on Object-Oriented Information Systems*, september 2002.
- [8] J. Ralyte. *Ingenierie des methodes a base de composants*. PhD thesis, Universite Paris I - Sorbonne, January 2001.
- [9] D. Rieu, J.P. Giraudin, and A. Conte. Pattern-based environements for information systems development. In *International Conference, The Science of Design*, Mars 2002.
- [10] C. Rolland, N. Prakash, and N. Benjamen. A multi-model view of process modelling. *Requirements Engineering Journal*, 4:169–187, 1999.
- [11] S. Si-said. *Proposition pour la modelisation et le guidage des processus d'analyse et de conception*. PhD thesis, Universite Paris I - Sorbonne, Fevrier 1999.
- [12] H. Storrlé. Describing process patterns with uml. In *ESWT*, September 2001.
- [13] K. van Slooten and B. Hodes. Characterizing IS development projects. In R.J. Welke S. Brinkkemper, K. Lyytinen, editor, *IFIP TC8, WG 8.1/8.2*, pages 29–44, August 1996.
- [14] B.G. Whitenack. RAPPeL: a requirement analysis process pattern language for oo development, 1995. http://www.bell-labs.com/user/cope/Patterns/Process/RAPPeL/rap_el.html.