

LABORATOIRE



INFORMATIQUE, SIGNAUX ET SYSTÈMES
DE SOPHIA ANTIPOLIS
UMR 6070

ORGANIZATION WIDE METHOD ENGINEERING

Isabelle MIRBEL

Projet EXECO

Rapport de recherche
ISRN I3S/RR-2006-11-FR

Mars 2006

RÉSUMÉ :

L'ingénierie des méthodes a pour vocation de trouver des solutions à la construction, l'amélioration et l'évolution des méthodes de développement. Différentes approches ont été successivement proposées pour proposer des méthodes d'aide au développement des systèmes d'information mais peu de recherches ont porté sur la personnalisation de telles méthodes quand elles sont utilisées comme des standards au niveau d'une entreprise. L'ingénierie des méthodes situationnelles, qui a pour but de proposer des techniques et des outils pour construire des méthodes répondant à la spécificité de chaque projet, ne répondent pas pleinement à ce besoin de personnalisation pour le projet à l'intérieur d'un cadre fixé au niveau de l'entreprise.

En fait, deux niveaux de personnalisation sont nécessaires: un premier qui doit permettre une adaptation au niveau de l'entreprise et un second niveau de personnalisation au niveau de chaque projet ou individu dans le projet. Notre travail fait partie d'un environnement complet pour l'ingénierie des méthodes situationnelles qui répond à ce double besoin.

Dans cet article, nous présentons le Reuse Frame, une structure ontologique qui permet de mieux centrer l'environnement méthodologique autour de l'utilisateur et de ses besoins en matière de méthodologie. Nous montrons comment cette structure ainsi que la définition d'une mesure de similarité et de proximité entre des fragments de méthodes permet aux ingénieurs méthode d'une part et aux ingénieurs d'application d'autre part de tirer un meilleur profit de l'ingénierie des méthodes situationnelles et de la réutilisation de fragments de méthode.

MOTS CLÉS :

Ingénierie des Méthodes, Réutilisation, Méthodes Situationnelles

ABSTRACT:

Method engineering aims at providing effective solutions to build, improve and support evolution of development methodologies. Different approaches have been successively proposed to provide suitable support to information system and software development but little research has focus on how to tailor such methodologies when used as organization-wide standard approaches. Situational method engineering, which aims at providing techniques and tools allowing to construct project-specific methodologies, do not fully answer the need of project customization inside the boundary of an organization-wide methodology. Indeed, two levels of customization are required: the first copes with the customization at the organization level and the second one deals with project/individual customization. Our work is part of a complete framework for situational method engineering which answers this twofold need.

In this paper we present the Reuse Frame, an ontological structure to help in better centering the methodological environment on method users and method engineering requirements. We also discussed our Similarity Metrics and Closeness Distance allowing method engineers and method users to better take advantage of situational method engineering and reuse perspectives.

KEY WORDS :

Method Engineering, Reuse, Situational Method Engineering

Organization Wide Method Engineering

Isabelle Mirbel

Laboratoire I3S, Les Algorithmes - Route des Lucioles, BP 121
F-06903 Sophia Antipolis, Cedex - France
isabelle.mirbel@unice.fr

Abstract: Method engineering aims at providing effective solutions to build, improve and support evolution of development methodologies. Different approaches have been successively proposed to provide suitable support to information system and software development but little research has focus on how to tailor such methodologies when used as organization-wide standard approaches. Situational method engineering, which aims at providing techniques and tools allowing to construct project-specific methodologies, do not fully answer the need of project customization inside the boundary of an organization-wide methodology. Indeed, two levels of customization are required: the first copes with the customization at the organization level and the second one deals with project/individual customization. Our work is part of a complete framework for situational method engineering which answers this twofold need. In this paper we present the Reuse Frame, an ontological structure to help in better centering the methodological environment on method users and method engineering requirements and in taking advantage of situational method engineering and reuse perspectives.

1 Introduction

Method engineering aims at providing effective solutions to build, improve and support evolution of development methodologies. Different approaches have been successively proposed to provide suitable support to information system and software development. Methodologies have been developed with a broad scope of situations in mind and look finally too generic to be applied as such in a specific project. Projects differ with respect to their development context, delivery, project team, deadline, etc. Almost every organization or project carries out tailoring in order to apply effectively best standard practices. Contributions, in the field of Situational Method Engineering (SME), aim at providing techniques and tools to construct project-specific methodologies [1,17,5]. SME approaches promote the construction and adaptation of new methods by assembling reusable *Method Fragments* stored in some method repository. The notion of method fragment represents the basic block for constructing 'on the fly' methods.

As it has been emphasized in [3], there is a lot of proposals on how to customize methodologies but little research has focused on how to tailor such methodologies when used as organization-wide standard approaches. SME do not fully answer the need of project customization inside the boundary of an organization-wide methodology. Indeed, two levels of customization are required:

the first copes with the customization at the organization level and the second one deals with project/individual customization. The work presented in this paper is part of a complete framework for SME which answers this twofold need [9].

In SME approaches, customization has mainly be thought of for the person in charge of the methodology, i.e. the method engineer (ME), in order to allow him/her to build new methodologies or improve existing ones [7,10]. But the person applying / using the methodology, i.e. the Method User (MU) (or application engineer) also needs dedicated customization means. Currently, MUs are required to know and understand the full methodology as well as all its concepts to be able to exploit the methodology, most of the time partially. Moreover, guidelines to manage and adopt process models are not detailed enough to support MUs through the meta-process understanding and MUs lack experience and ability to establish 'home grown' development methodologies or to tailor existing methodologies [15]. There is a tension between the 'method-in-concept' (the methodology as formalised in manual) and the 'method-in-action' (as interpreted by MUs) [4]. Experiments show that it all has negative effects and discredits methodologies. MUs need a dedicated, targeted, easy and fast access to the methodology. The individual customization level that we provide in our approach is an answer to this need.

Currently, MUs prefer lightweight processes/methodologies to heavyweight ones because they feel more implicated. Lightweight methodologies increase MUs involvement on the contrary of heavyweight methodologies where the only significant choices are made by MEs. Feedback from MUs shows that methodologies are seen as too prescriptive and too rigid [13]. But lightweight processes are most of the time empirical processes derived by categorizing observed inputs and outputs, and by defining meaningful controls to keep the process within prescribed bounds [12]. In empirical method modeling, models are strictly based on experimentally obtained input/output data, with no recourse to any laws concerning the fundamental nature and properties of the application to develop, the project or the methodology itself. Therefore it makes it very difficult to transpose it from one organization to another. SME could be better accepted by MUs if involvement means are also provided: Customization at personal level is a way to involve MUs. But it requires that MUs express and communicate their specific need in terms of Method Requirement (MR), which looks difficult for them [16]. Our proposal helps MUs during these activities.

SME deals with the building or customization of specific methodology which will last for a relatively long period of time because of the commitment and investment it requires, especially if organization-wide. On the other hand, the constant evolution of techniques, mechanisms and technologies applied to develop software or information system requires support for methodological evolution. Moreover, MRs change during the project lifetime, and this kind of evolution has not at all be handled by SME, which is anchored in the basic assumption that MR do not change during the project lifetime [16]. Some current approaches deal with

methodology evolution: method rationale and evolutionary method engineering [15], but they focus on the methodological content evolution while we try to provide solution to support MR evolution.

Finally, organization-wide SME requires to capitalize and share knowledge about method engineering. It is recognized as important to benefit from the experiences acquired during the resolution of previous problems through reuse and adaptation mechanisms. Existing environments in the field of software engineering are dedicated to experts having strong knowledge about fragment repositories and the way they are organized. Such repositories are not dedicated to users who are not experts and for who the major interest of such environments would be to be assisted in their search for the most suitable fragments [18]. Recent approaches combine user intention and application domain information, as well as natural language techniques [18]. But method engineering is not application-domain dependent and these approaches therefore are not suitable for method engineering reuse. With regards to software development, reuse has been widely studied from the product point of view [6,14], but it is now also a challenging issue to handle it from the process point of view. In [19], a framework for component reuse in meta-modelling based software development is presented. It synthesizes different types of reuse in system development and requirement engineering processes. The authors emphasize the fact that reuse of system development knowledge in terms of methods and component methods must be considered. With regards to their framework, we propose an approach to deal with functional reuse along all the activities within system development and requirement engineering processes. Our focus is on method engineering activities and knowledge and we cover the abstraction and selection steps of the reuse process [2].

The remainder of the paper is organized as follows. In the next section we explain why organization wide method engineering requires a user-centric approach and how we answer this need. Section 3 develops a polymorphic and scalable structure to support method engineering knowledge capitalization and sharing : the *Reuse Frame*. Section 4 discusses models and metrics to support the selection of meaningful method fragments through customization and configuration activities. Section 5 concludes the paper and discusses future research.

2 User-centric Method Engineering

In this paper, we present a polymorphic and scalable structure to help in better centering the methodological environment on MEs and MUs needs and allowing to take advantage of SME and reuse perspectives. Our proposal aim is to extend the scope of SME by allowing individual MUs to adapt their way of applying methodological guidelines on the fly while still obeying to some degree to the organization-wide specific method defined by MEs. Our approach is dedicated to MEs as well as MUs: MEs have the knowledge about method engineering and about the organization/company they belong to. They need means (i) to manage the knowledge about MR, (ii) to maintain the method fragment repository and

(iii) to query the fragment repository to build the organization-wide specific method. MUs use (part of) the method in their daily work: they need means to query the organization-wide specific method and the method fragment repository to find suitable methodological guidelines with regards to their job. Indeed, we support two levels of customization:

- At organization/company level we support customization for MEs through the process of building the organization-wide specific method.
- At individual level, we support customization for MUs through their personal configuration of the organization-wide specific method.

More precisely, we talk about *method customization* to indicate the building of the organization-wide specific method and about *method configuration* to indicate the process of selecting meaningful method fragments inside the organization-wide specific method. In [16], method configuration is also defined as a special kind of method engineering where one specific method is the starting point for tailoring.

While it is obvious that an organization-wide specific method is required to allow everyone to share working ways, MEs and MUs perform different jobs and therefore require specific access to the method knowledge.

Up to now, method fragments have been thought of to support the building of new and better-adapted methods. Component reuse has been studied in order to capitalize knowledge about the application to develop (with the help of design patterns). But poor attention has been given to application domain independent reuse and knowledge about MR. We believe reuse as well as method building have to be driven by MR knowledge. A method fragment repository should be seen as a repository of experiences about method engineering and means have to be provided to maintain method engineering knowledge in a pragmatic oriented way in order to focus on MU as well as ME daily difficulties.

In this context, the *Reuse Frame* we propose in our approach is an alternative way to organize method fragments in order to improve their reusability. It aims at capturing relationships among information system aspects to facilitate navigation through related method fragments in an ontological way. Reuse is exploited by MU to retrieve method fragments and by ME to qualify method fragments. When building a method, MEs search for method fragments in the repository in order to assemble them into the new organization-wide specific method. During this retrieval task, they behave as MUs. Therefore, in the remaining of this paper we will use the term of MU to talk about MU and also ME when searching method fragments in the repository.

3 The Reuse Frame

Critical knowledge about method engineering should be taken into consideration in addition to structural knowledge and keywords when providing support to method customization and method configuration. From our point of view, current classification and retrieving means are not fully suitable because they are

supported by the structure of the method they are part of (i.e. the relationships provided by the meta-model of the method to link artifacts among them). Recent work on method component reuse combines user intention and application domain information in order to provide alternative and richer means to organize and retrieve components [18]. But again, domain information does not look like the most suitable information to support method engineering reuse that should be possible across different application domains. The only knowledge that will be understandable by everyone (that is to say knowledge which is neither application domain oriented nor organization-specific method oriented) is the method engineering knowledge. We distinguish 3 main families of critical aspects about method engineering [11,8] : *human*, *organizational* and *application domain*.

- The *Human* perspective allows to specify the kind of stakeholder the method fragment is dedicated to (developer, designer, product definition responsible, test manager, etc.). It also allows to qualify his/her expertise: expert, beginner, etc. An example of *Human* aspect is **Expert analyst**.
- The *Organizational* perspective groups aspects related to contingency factors, project characteristics, goals and assumptions, as well as system engineering activities. With regards to the *organizational* dimension, we started from the work of K. van Slooten and B. Hodes providing elements to characterize projects [11]. **Real User Involvement** is an example of meaningful contingency factor. **Standard Project Organization** and **Strong Project Tracing** are examples of project-related aspects.
- The *Application Domain* perspective [8] reassembles aspects about the kind of application to develop. With regards to this 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 [8]. **Inter-organization application (B2B)** and **Application to develop includes a database** are examples of application-domain related aspects.

3.1 Aspect Refinement

Aspects are shared by all the MEs and MUs and give support to knowledge capitalization and reuse. Indeed, an aspect represents a point of view on the method to allow MEs to better drive MUs on their way to configure and exploit as much as possible the organization-wide specific method. Aspects may be successively refined. We distinguish different kinds of refinement relationships between aspects: refinement into more specific aspects, refinement into more specific and classified aspects, refinement into more specific and exclusive aspects.

An example of refinement into more specific aspects without any other constraint is the **Application Technology** aspect which is for instance refined into **Application to develop includes a Graphical User Interface** and **Application to develop includes databases**.

The refinement into more specific and *classified* aspects allows to specify an order among the different aspects at a same refinement level. This classification

information may be helpful when looking for method fragments associated with a specific aspect to look also at method fragments associated with aspects classified previous or next the aspect under consideration. **Code Reuse**, for instance, may be refined into **Weak Code Reuse**, **Medium Code Reuse** and **Strong Code Reuse**. When looking at method fragments qualified with the help of **Medium Code Reuse**, it may be also interesting to look at method fragments associated with **Strong Code Reuse** or **Weak Code Reuse**, especially if no fragment qualified by **Medium Code Reuse** are found.

The refinement into *exclusive* aspects prevents (i) MEs from qualifying fragments through incompatible aspects and (ii) MUs from qualifying their MR through incompatible aspects. **Inter-organization Application**, **Intra-organization Application** and **Organization-customer application** are examples of exclusive refinements of the **Application Type** aspect: A method fragment providing guidelines on how to deal with an **Inter-organization Application** can't be also suitable for an **Intra-organization Application** or **Organization-Customer Application**.

3.2 Reuse Frame Structure

In our approach, aspects are specified with the help of a dedicated tree, called the *Reuse Frame*. In the *Reuse Frame*, the root node is mandatory, as well as the 3 main aspects: **Human**, **Organizational** and **Application** domain. Nodes are refined through the refinement relationships previously presented. Nodes close to the root node correspond to general aspects while nodes close to leaf nodes (including leaf nodes) correspond to precise aspects.

A classification relationship requires a starting node and at least two ending nodes. A node is exclusive with regards to at least another one. Therefore, each node may have 0 or more than 1 starting edges. All starting edges must be from the same kind: if different kinds of edges are required, it means that a intermediary node is needed to clearly group the edges of the same kind. Figure 1 shows examples of well-formed refinement relationships.

An *aspect* is a node different from the **Root**, **Organizational**, **Application Domain** and **Human** nodes (which are too generic to be useful to qualify method fragments and MR) and which starting edges are not labeled as exclusive (otherwise, only ending exclusive nodes may be considered as valid aspects). Figure 1 shows part of a well-formed *Reuse Frame*. To see the full *Reuse Frame*, please refer to the Reuse Frame Web Page and [9].

Inclusion between aspects: An aspect **a1** is included in an aspect **a2** if the path from the root node to **a1** is a sub-path of the path from the root node to **a2**. In Figure 1, **High complexity** *includes* **Contengency factor**.

Precedence between aspects: An aspect **a1** is *previous* an aspect **a2** if they have the same upper node and the classification rank of *a1* is inferior to the

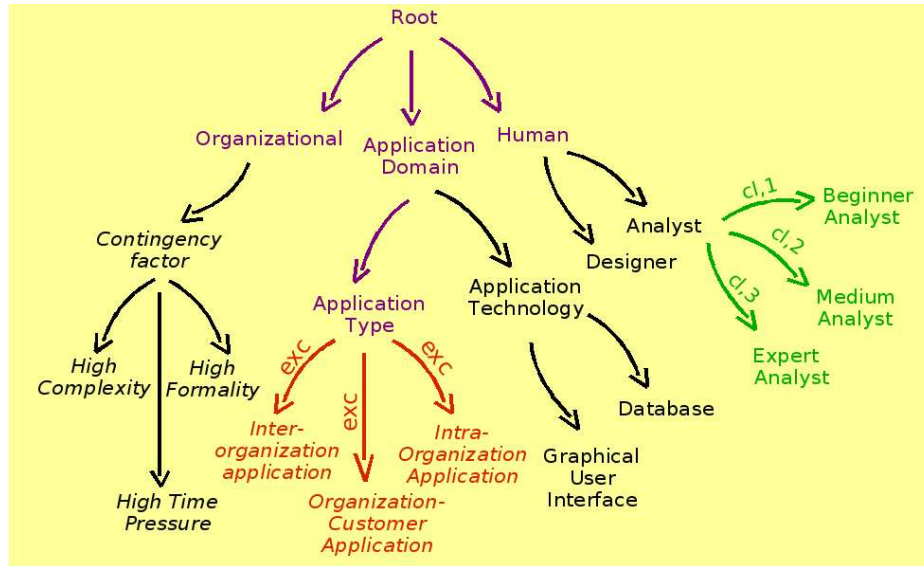


Fig. 1. Part of the Reuse Frame

classification rank of $a2$: $cl(a1) < cl(a2)$. $a1$ is *next* $a2$ if $a1$ and $a2$ have the same upper node and the classification rank of $a2$ is inferior to the classification rank of $a1$: $cl(a1) > cl(a2)$. In Figure 1, Expert Analyst is *next* Medium Analyst.

Compatibility between aspects: Compatible aspects, if they are not included one in the other, do not share in their path (from the root node) a node with exclusive leaving edges. In Figure 1, Inter-organization Application and Intra-organization Application are *incompatible* aspects. High Formality and High Complexity are *compatible* aspects.

The aspects defined in the *Reuse Frame* aim at supporting capitalization about method engineering knowledge. In the following we will discuss the notion of *Method Fragment Reuse Context* and *Method User Reuse Situation* which aim at qualifying method fragments and specifying MR need with regards to the aspects defined in the *Reuse Frame*.

4 Supporting Organization-Wide Method Engineering

4.1 Method Fragment Reuse Context

In SME approaches, method fragments are stored in dedicated repositories in order to be retrieved and assembled into new methods. Method fragments need to be qualified by specific slots to be retrieved in the organization repository

(i) by the MEs during the method building process and (ii) by MUs during the method configuration process. Retrieval may be supported by keyword matching on specific slots. But we believe it is not sufficient and MR knowledge should be exploited too. Therefore, we propose to enrich method fragment description with a *Method Fragment Reuse Context*, abbreviated *Reuse Context* in the following. It aims at clearly specifying the reuse context of the method fragment with the help of the *Reuse Frame*. The *Reuse Context* RC_f of a method fragment f is defined as the set C_f of at least one aspect c_i selected among thus provided in the *Reuse Frame* RF . Aspects of C_f must be compatible among them and can't be included one in each other.

$$RC_f = \langle C_f \rangle \text{ where } C_f = (c_1, \dots, c_n) \text{ with } c_i \in RF$$

Usually, method fragments providing general guidelines are qualified with the help of less refined aspects (i.e. corresponding to nodes close to the root node). Method fragments providing specific guidelines are qualified with the help of more refined aspects (i.e. corresponding to nodes close to the leaf nodes or leaf nodes themselves). An example of *Reuse Context* for a method fragment entitled **Behavior Investigation** is presented in Figure 2. For more information about the specification of the method fragment body, please refer to [10].

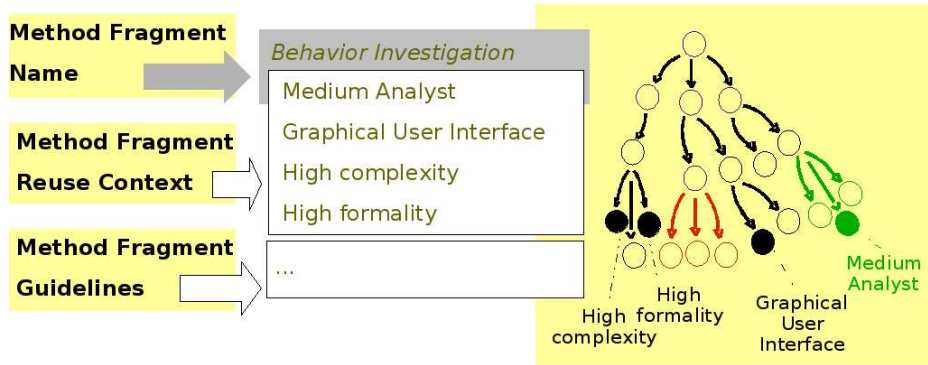


Fig. 2. Example of Method Fragment Reuse Context

The *Reuse Context* allows MEs to abstract reusable knowledge from method fragments.

4.2 Method User Reuse Situation

To extend the scope of SME, we provide MUs with means to query the method fragment repository or the organization-wide specific method to find suitable

methodological guidelines with regards to their need. It is supported by the *Method User Reuse Situation*, abbreviated *Reuse Situation* in the following. A *Reuse Situation* allows MUs to express their MR need by selecting aspects among thus stored in the *Reuse Frame*. In the *Reuse Situation*, in addition to the pertinent aspects, called *necessary aspects*, MUs may give *forbidden aspects*, that is to say aspects they are not interested in. It could be helpful in some cases to be sure the method fragments including these (forbidden) aspects will not appear in the retrieved set of method fragments answering the methodological need. A *Reuse Situation* is indeed a set of at least one aspect to specify necessary features and a set of aspects to specify forbidden features. All aspects must be compatible among them inside each set and the following constraints must hold between necessary and forbidden aspect sets:

- No common aspects between necessary and forbidden aspects;
- No inclusion between necessary and forbidden aspects: It is not possible by definition to find two aspects included one in the other in the same *Reuse Context*;
- No incompatibility among necessary and among forbidden aspects.

The *Reuse Situation* RS of a MU u is defined as a n-uplet $RS_u = \langle NC_u, FC_u \rangle$ where :

- NC_u is the set of at least one necessary aspects qualifying the MR need of the MU u . $NC_u = \{c_{NC_{u_1}}, \dots, c_{NC_{u_n}}\}$ with $c_{NC_{u_i}} \in RF$;
- FC_u is the set of forbidden aspects also qualifying the MR need of the MU u . $FC_u = \{c_{FC_{u_1}}, \dots, c_{FC_{u_m}}\}$ with $c_{FC_{u_j}} \in RF$.

If MUs search for general guidelines, they should select necessary aspects among the less refined aspects of the *Reuse Frame*, that is to say aspects corresponding to nodes close to the root node. On the contrary, if they search for specific guidelines, they may specify their need by selecting in the *Reuse Frame* more refined aspects, that is to say aspects corresponding to nodes close to the leaf nodes or leaf nodes themselves. An example of *Reuse Situation* is shown in Figure 3.

By comparing the *Reuse Context* associated to each method fragment stored in the repository with the *Reuse Situation* given by the MU, meaningful method fragments are selected. This selection may also be done inside the method fragment set corresponding to the organization-wide specific method instead of the whole repository. A *Reuse Context* may not fully match a *Reuse Situation*: Only some aspects may be shared by both of them. Therefore, we introduce the *Similarity Metrics* to quantify the matching between a *Reuse Context* and a *Reuse Situation*.

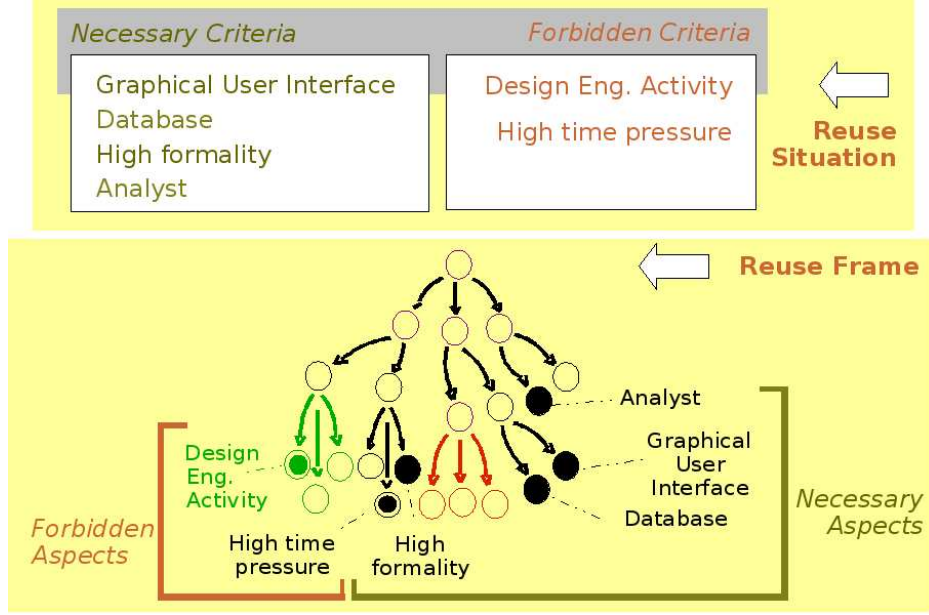


Fig. 3. Example of Method User Reuse Situation

4.3 Similarity Metrics

The *Similarity Metrics* is based on (i) the number of common aspects between the necessary aspects from the *Reuse Situation* and the *Reuse Context*, (ii) the number of common aspects between the forbidden aspects from the *Reuse Situation* and the *Reuse Context*, (iii) the number of necessary aspects in the *Reuse Situation*:

$$sm(RC_f, RS_u) = \frac{\sum_{i=1..n} d(c_{NC_{u_i}}, C_f) - \sum_{j=1..m} d(c_{FC_{u_j}}, C_f)}{card(NC_u)}$$

where RS_u is the *Reuse Situation*, $NC_u = \{c_{NC_{u_1}}, \dots, c_{NC_{u_n}}\}$ its necessary aspect set, $FC_u = \{c_{FC_{u_1}}, \dots, c_{FC_{u_m}}\}$ its forbidden aspect set. RC_f a *Reuse Context*, C_f its set of aspects and $d(c, C)$ a distance between an aspect and a set of aspects defined as follows: If $c \in C$, $d(c, C) = 1$, else $d(c, C) = 0$.

A positive value of $sm(RC_f, RS_u)$ indicates that there are more necessary aspects than forbidden ones in the *Reuse Context* RC_f with regards to the *Reuse Situation* RS_u . On the contrary, a negative value of $sm(RC_f, RS_u)$ indicates that there are less necessary aspects than forbidden ones. The perfect adequation is represented by the value 1 and the worst situation by the value:

$$sm(RC_f, RS_u) = -\frac{card(FC_u)}{card(NC_u)}$$

4.4 Method Fragment Closeness

In addition to fragments which *Reuse Context* partially match the *Reuse Situation*, it may also be interesting to retrieve *close* method fragments. For this purpose, we exploit the relationships among aspects in the *Reuse Frame* (refinement, classified refinement).

Method fragments including more general, more specific, previous or next aspects in their *Reuse Context*, with regards to the *Reuse Situation*, are considered as *close* method fragments. Method fragments including more specific aspects in their *Reuse Context* may be of interest because they usually provide more specific guidelines. They may better cover part of the methodological problem the MU is interested in. If one searches for instance for method fragments matching the **Code Reuse** aspect, he/she may also be interested by method fragments matching the **Weak Code Reuse**, **Medium Code Reuse** aspect and **Strong Code Reuse** as shown in Figure 4. MU may also be interested in method fragments qualified by more general aspects because they may provide more general-purpose guidelines which could also be of interest. In the same way the classification dimension of refinement relationships may be exploited to enlarge the set of method fragments retrieved with previous and next aspects. Examples are shown in Figure 4.

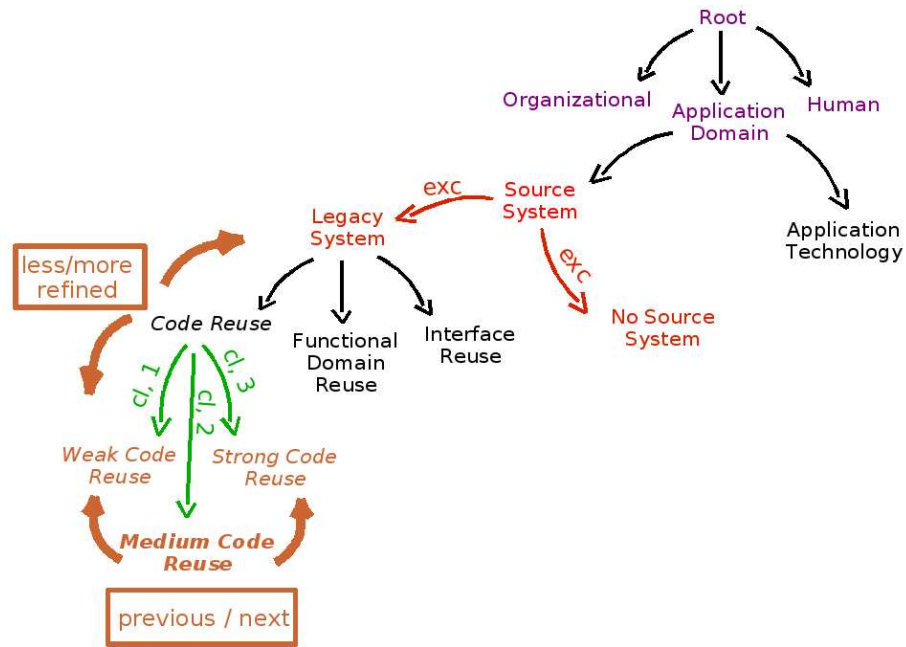


Fig. 4. Example of extension through refinement relationships

Exploring *Reuse Frame* refinement relationships to enlarge the query to close method fragments may also be interesting with regards to *forbidden aspects*. Indeed, enlarging the set of forbidden aspects to more general ones means to forbid full branches of the *Reuse Frame* and enlarging the set of forbidden aspects to more specific aspects means to forbid method fragment associated to too specific aspects, most probably qualifying method fragments providing too specific guidelines. In the same way, enlarging the set of forbidden aspects to aspects previous or next the aspects under consideration (thanks to classified refinement relationships) means to avoid retrieving method fragments which scope overcomes the aspects given by the MU. Tuning the selection by allowing to retrieve or exclude close method fragments provides a way for the MU to reduce or enlarge the number of method fragments retrieved.

When extending query to close method fragments, the distance with regards to the reference situation (i.e. with regards to the aspects given by MUs) has to be quantified to better let them understand how much the presented method fragments match their MR need.

4.5 Quantifying method fragment closeness

To quantify the distance between a *close* method fragment and the reference situation, the *Similarity Metrics* is computed with necessary or forbidden aspects extended to more generic, specific, previous or next aspect sets. Indeed the computation of the distance $d(c, C)$ (between an aspect and a set of aspects) introduced in the *Similarity Metrics* definition is extended by being based on the computation of a *Closeness distance* (between two aspects). We propose 4 distances to qualify the closeness with more generic, more specific, previous and next aspects. A perfect matching between two aspects leads to the value 1 of the *Closeness distance*, which tends to 0 as far as the aspects look distant as explained in the following.

MORE GENERIC ASPECTS : The *Closeness distance* between an aspect **a1** and an aspect **a2** more *generic* than **a1** is computed according to the ratio between the number of node between **a1** and **a2** and the number of node between **a1** and the root node. Let l_{a_1} be the number of nodes in the path from the **root** node to the node corresponding to the aspect a_1 under consideration in the *Reuse Situation RS*. Let l_{a_2} be the number of nodes in the path from the **root** node to the node corresponding to the aspect a_2 under consideration in the *Reuse Context RC*. a_2 belongs to the path from the root node to a_1 . The closeness distance between a_1 and a_2 is:

$$cd(a_2, a_1) = 1 - (l_{a_1} - l_{a_2})/l_{a_1}$$

SPECIFIC ASPECTS : The *Closeness distance* between an aspect **a1** and an aspect **a2** more *specific* than **a1** is computed according to the ratio between the number of node between **a1** and **a2** and the number of node in the longest path from **a1**

to the deepest leaf-node. Let l_{a_1} be the number of nodes in the path from the **root** node to the node corresponding to the aspect a_1 under consideration in the *Reuse Situation RS*. Let l_{a_2} be the number of nodes in the path from the **root** node to the node corresponding to the aspect a_2 under consideration in the *Reuse Context RC*. Let $l_{a_{max}}$ be the number of nodes of the longest path from a_1 to a leaf node a_{max} . a_1 and a_2 belong to the same path from the root node to a_{max} . The closeness distance between a_1 and a_2 is:

$$cd(a_2, a_1) = 1 - (l_{a_2} - l_{a_1}) / (l_{a_{max}} - l_{a_1} + 1)$$

PREVIOUS ASPECTS : The *Closeness distance* between an aspect **a1** and an aspect **a2** *previous* **a1** is computed according to the ratio between the number of node between **a1** and **a2** and the number of node previous **a1**. Let a_1 be the aspect under consideration in the *Reuse Situation RS* and cl_{a_1} its classification rank. Let a_2 be the aspect under consideration in the *Reuse Context RC* and cl_{a_2} its classification rank. a_1 is *next* a_2 . The closeness distance between a_1 and a_2 is:

$$cd(a_2, a_1) = 1 - (cl_{a_1} - cl_{a_2}) / cl_{a_1}$$

NEXT ASPECTS : The *Closeness distance* between an aspect **a1** and an aspect **a2** *next* **a1** is computed according to the ratio between the number of node between **a1** and **a2** and the number of node next **a1**. Let a_1 be the aspect under consideration in the *Reuse Situation RS* and cl_{a_1} its classification rank. Let a_2 be the aspect under consideration in the *Reuse Context RC* and cl_{a_2} its classification rank. Let $cl_{a_{max}}$ be the highest classification rank among the aspects next a_1 . a_1 is *previous* a_2 which is *previous* a_{max} . The closeness distance between a_1 and a_2 is:

$$cd(a_2, a_1) = 1 - (cl_{a_2} - cl_{a_1}) / (cl_{a_{max}} - cl_{a_1} + 1)$$

Figure 5 shows part of the *Reuse Frame* and examples of closeness distances.

EXTENSION COMBINATION : Thanks to the previously defined closeness distance, the *Similarity Metrics* is expendable. When the aspect which is present in the *Reuse Situation* of the method fragment under consideration is not identical to one of the aspects of the *Reuse Situation* but only close to it, the closeness distance is used instead of the value 1 in the computation of the *Similarity Metrics*.

5 Concluding Remarks and Future Works

In this paper we presented an approach to help in supporting and personalizing access to method fragments for method engineers when building the organization-wide specific method and method users when configuring the

More generic aspects
 $cd(a, d) = 1 - (4-1)/4 = 0,25$
 $cd(b, d) = 1 - (4-2)/4 = 0,5$
 $cd(c, d) = 1 - (4-3)/4 = 0,75$

More specific aspects
 $cd(e, d) = 1 - (5-4)/3 = 0,66$
 $cd(f, d) = 1 - (6-4)/3 = 0,33$

Previous aspects
 $cd(1, 3) = 1 - (3-2)/3 = 0,66$
 $cd(2, 3) = 1 - (3-1)/3 = 0,33$

Next aspects
 $cd(6, 3) = 1 - (6-3)/4 = 0,25$
 $cd(5, 3) = 1 - (5-3)/4 = 0,5$
 $cd(4, 3) = 1 - (4-3)/4 = 0,75$

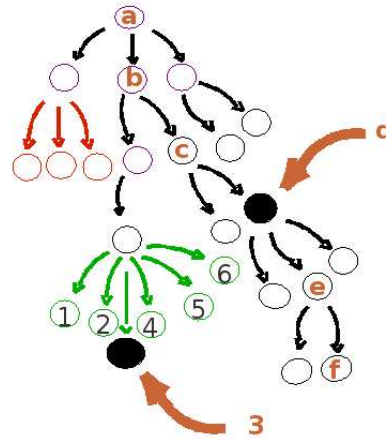


Fig. 5. Closeness distance between aspects

organization-wide specific method. This work is part of a complete framework for situational method engineering in which we consider two perspectives: (1) organization-wide specific method engineering which aims at satisfying specific organization method requirement and (2) user-specific method engineering dedicated to satisfy requirements of an individual method user. We believe that these two perspectives are complementary and should be combined in order to support the information system development process at hand. In this paper, we focussed on the *Reuse Frame* providing an innovative way to organize method fragment with regards to method requirement knowledge. We also discussed our *Similarity Metrics* and *Closeness Distance* allowing to extend the method fragment selection process to fragments partially matching method users need and to *close* method fragments. Our contribution aims at taking advantage of situational method engineering and reuse perspectives to better center the methodological environment on method users. To evaluate this approach by applying it in real information system development projects is our current preoccupation. A first prototype has already been developed.

In the future, we would like to improve our approach by extending it to other kinds of stakeholders by refining the notions of method engineer and method user. We would also like to enrich it by capturing and exploiting experiences, practices and feedback information on the current way of performing method engineering in the organization and this way better supporting the two next steps of a reuse process: specialization and integration.

- [1] A.F. Harmsen – **Situational Method Engineering** – *Moret Ernst Young*, 1997.
- [2] A.G. Sutcliffe, N.A.M. Maiden – **Supporting Component Matching for Software Reuse.** – CAiSE, 1992, pp. 290-303.

- [3] B. Fitzgerald, N.L. Russo, T. O’Kane: – **Software development method tailoring at Motorola** – Communications of the ACM, 46(4), 2003, pp. 64-70.
- [4] B. Lings, B. Lundell – **Method-in-Action and Method-in-Tool: Some Implications for CASE** – ICEIS 2004, Porto, Portugal, April, 2004.
- [5] C. Rolland, V. Plihon, J. Ralyté – **Specifying the Reuse Context of Scenario Method Chunks** – International Conference on Advanced Information System Engineering, 1998.
- [6] E. Gamma and R. Helm and R. Johnson and J. Vlissides – **Design Patterns: Elements of Reusable Object-Oriented Software** – Addison-Wesley Publishing Company, 1995.
- [7] F. Harmsen, S. Brinkkemper, J. L. Han Oei – **Situational method engineering for informational system project approaches** – IFIP WG8.1 Working Conference on Methods and Associated Tools for the Information Systems Life Cycle, 1994.
- [8] I. Mirbel and V. de Rivieres – **Adapting Analysis and Design to Software Context: The JECKO Approach** – OOIS 2002, Montpellier, France, September, 2002, pp. 223-228.
- [9] I. Mirbel, J. Ralyte – **Situational method engineering: combining assembly-based and roadmap-driven approaches** – Requirement Engineering Journal, 11(1), 2006, pp. 58-78.
- [10] J. Ralyte – **Ingenierie des methodes a base de composants** – Université Paris I - Sorbonne, January, 2001.
- [11] K. van Slooten and B. Hodes – **Characterizing IS Development Projects** – IFIP TC8, WG 8.1/8.2, August, 1996, pp. 29-44.
- [12] L. Rising, N.S. Janoff – **The Scrum software development process for small teams** – IEEE Software, 17(4), 2000, pp. 26-32.
- [13] M. Bajec, D. Vavpotic, M. Kirsper – **The scenario and tool-support for constructing flexible, people-focused system development methodologies** – ISD 2004, Vilnius, Lituania, September, 2004.
- [14] M. Fowler – **Analysis Patterns: Reusable Object Models** – Addison-Wesley Publishing Company, 1996.
- [15] M. Rossi and B. Ramesh and K. Lyytinen and J. Tolvanen – **Managing evolutionary method engineering by method rationale** – Journal of the association for information systems, 5(9), 2004, pp. 356-391.
- [16] P.J. Agerfalk, F. Karlsson – **Method configuration: adapting to situational characteristics while creating reusable assets** – Information and software technology, 46(9), 2004, pp. 619-633.
- [17] S. Brinkkemper, M. Saeki, F. Harmsen – **Assembly Techniques for Method Engineering** – International Conference on Advanced Information Systems Engineering, 1998.
- [18] V. Pujalte, P. Ramadour – **Réutilisation de composants: un processus interactif de recherche** – Majecstic’05, 2004.
- [19] Z. Zhang, K. Lyytinen – **A Framework for Component Reuse in a Metamodelling-Based Software Development** – Requirement Engineering Journal, 6(2), 2001, pp. 116-131.