A REUSE FRAME FOR METHOD ENGINEERING

Isabelle MIRBEL

Projet EXEKO

Rapport de recherche
ISRN I3S/RR–2006-05–FR

Février 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 standard 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 au spécificité de chaque projet, ne répondent pas pleinement à ce besoin de personnalisation pour le projet à l’intérieur d’une 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 polymorphe qui aide à mieux centré l’environnement méthodologique autour de l’utilisateur et de ses besoins en matière de méthodologie. Cette structure permet également de mieux tirer partie de l’ingénierie des méthodes situationnelles et de la réutilisation de fragments de méthode. Le Reuse Frame permet l’accès personnalisé à la méthodologie aux ingénieurs d’application lorsqu’ils construisent la méthode spécifique à l’entreprise et aux utilisateurs de la méthode spécifique lorsqu’ils l’utilisent pour leur travail quotidien.

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, a polymorphic and scalable structure to help in better centering the methodological environment on method users and method requirements and allowing to take advantage of situational method engineering and reuse perspectives. It helps in supporting and personalizing access to methodology for method engineers when customizing methods into organization-specific methods and to method users when configuring an organization-specific method to answer their personal method requirement need.

KEY WORDS :

Method Engineering, Reuse, Situational Method Engineering
A Reuse Frame for Method Engineering

Isabelle Mirbel
Laboratoire I3S, Les Algorithmes - Route des Lucioles, BP 121
F-06903 Sophia Antipolis, Cedex - France

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, a polymorphic and scalable structure to help in better centering the methodological environment on method users and method requirements and allowing to take advantage of situational method engineering and reuse perspectives. It helps in supporting and personalizing access to methodology for method engineers when customizing methods into organization-specific methods and to method users when configuring an organization-specific method to answer their personal method requirement need.

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 [7,10]. 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 fields of Situational Method Engineering (SME), aim at providing techniques and tools allowing to construct project-specific methodologies [22,1,33,9]. 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 represent the basic block for constructing 'on the fly' methods.

As it has been emphasized in [3], there are hundreds of methodologies that have been developed but little research has focus 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. Our work is part of a complete framework for SME which answers this twofold need [20].

In current SME approaches, customization have 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 [24,8,32,13,15,11,29,21]. But the person applying the methodology, i.e. 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 [28]. There is a tension between the ‘method-in-concept’ (the methodology as formalised in manual) and the ‘method-in-action’ (as interpreted by the MU) [2,4]. Experiments show that it all has negative effects and discredits methodologies [26]. MUs need a dedicated, targeted, easy and fast access to the methodology. 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 ME. Feedback from users shows that methodologies are seen as too prescriptive and too rigid [26]. 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 [25]. But 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, to allow an organization to take advantage of the methodology build in another organization. There is a need for means to involve MUs in the process, for instance by the way of customization. And finally, it is difficult for MUs to explicitly specify the required situational methodology and communicate these methodological requirements (MR) [31]. Therefore, there is a need for means dedicated to MUs to express easily their MRs.

Organization-wide SME deals with the building or customization of an organiza-
tion-specific methodology which will last for a relatively long period of time because of the commitment and investment it requires. On the other hand, the constant evolution of techniques, mechanisms and technologies applied to develop a software or an information system requires methodological evolution means. 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, as it is explained in [31]. Current approaches to deal with methodology evolution are method rationale and evolutionary method engineering [28]. But these approaches focus on method 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 [6]. With regards to software development, reuse has been widely studied from the product point of view [12,27], but it is now also a challenging issue to handle it from the process point of view [14,30]. Efforts have already been made in the fields of method engineering to provide efficient classification and retrieving means to store and select Method Fragments. But method reuse is currently based on structural relationships among Method Fragments (specialization, composition, alternative, ...) and keyword matching [5]. These techniques are not well-adapted to method engineering knowledge and especially MR. Moreover, existing environments are dedicated to experts having strong knowledge about fragment repositories and the way they are organized. Once again, such repositories are not dedicated to MUs who are not experts and for which the major interest of such environments would be to be assisted in their search for the most suitable fragments [35,34]. Recent approaches [35,34] combine user intention and application domain information, as well as natural language techniques. From our point of view, these techniques do not fully exploit MR knowledge.

In this context and to answer all these needs, we propose an environment for organization-wide SME, dedicated to MEs and MUs and supporting MR knowledge capitalization and evolution.

2 User-centric Method Engineering

We propose a polymorphic and scalable structure to help in better centering the methodological environment on MEs and MUs requirements and to better take advantage of SME and reuse perspectives. Therefore, we try to extend the scope of SME by allowing individual team members to adapt their Method Fragments on the fly while still obeying to some degree to the organization-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 to manage the knowledge about MR as well as Method Fragment
repository. MUs use (part of) the method in their daily work: they need means to query the Method Fragment repository to find suitable methodological guidelines with regards to their work. Indeed, we support 2 levels of customization in our approach:

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

We talk about method customization to indicate the building of the organization-specific method and about method configuration to indicate the process of selecting meaningful Method Fragments inside the organization-specific method. In [31], 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-specific method is required to allow everyone to share working ways, ME and MU perform different works and therefore require specific access to the method knowledge. Means have to be provided to let them express MR and retrieve meaningful Method Fragments with regard to their needs. Moreover, specific organization means for Method Fragments have also to be provided.

Method Fragments have been thought of in order 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 knowledge about MR. We believe reuse as well as method building have to be driven by MR knowledge [16]. Method Fragment repositories should be seen as repositories of experiences about Method Engineering and means have to be provided to maintain and exploit them in a pragmatic oriented way in order to focus on MU daily difficulties.

In this context, the Reuse Frame we propose is an alternative way to organize Method Fragments in order to improve their reusability (to be applied by MU to configure method or by ME to customize methods). It aims at capturing relationships among Information System Development (ISD) aspects to facilitate navigation through related Method Fragments in an alternative way to usual navigation means (decomposition relationships). We combine keywords and ontological based retrievals in order to support Method Engineering knowledge reuse. Moreover, we provide means to combine several characteristics and capture many usage situations in a rich way. We explicitly provide means to specify requirements on a situational method.

3 The Reuse Frame

We believe critical knowledge about ISD [5] should be taken into consideration in addition to structural knowledge and keywords when looking at means to store and organize Method Fragment in a reuse or assembly purpose. This knowledge can be used to qualify each Method Fragment in the repository. We
distinguish 3 main families of critical aspects [23,17,19]: human, organizational and application domain related aspects.

– The Human perspective allows to specify the kind of stackholder 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 ISD projects [23]. Real User Involvement is an example of contingency factor. Standard Project Organization and Strong Project Tracking are examples of project-related aspects. Business Modeling is an example of aspect associated with system engineering activities.

– The Application Domain perspective [18] reassembles aspects about the kind of application, the presence of a legacy system and the technology required by the application to develop. 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 [18]. Inter-organization application (B2B) is an example of application type. Strong Code Reuse is an example of aspect associated with development on top of a legacy application. Application to develop includes a database is an example of aspect describing the technology used by the application.

3.1 Aspect Refinement

Indeed, an aspect represents an ISD point of view on the method to allow MEs to better drive MUs on their way to apply the organization-wide situational method and to provide them with richer means to broadcast way of working than deliverables, as it is usually the case. Aspects are shared by all the MEs and MUs and give support to knowledge capitalization, sharing and reuse. We distinguish different levels of abstraction in the MR knowledge: Weak Code Reuse is for instance a more precisely defined aspect than the one specifying that the development is made on top of a Legacy System. Therefore, the MR knowledge is specified in terms of node which are successively refined to represent the different abstraction levels.

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 constraints 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 some order among the different aspects at a same refinement level. This classification information may be helpful when retrieving Method Fragments associated with one given aspect to look also at Method Fragments associated with the 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. And when looking at Method Fragments qualified with the help of Medium Code Reuse, it may also be interesting to look at Method Fragments associated with Strong Code Reuse and Method Fragments associated with Weak Code Reuse, especially if no fragment associated with Medium Code Reuse are found.

The refinement into exclusive aspects is also another useful kind of relationship. It avoids (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 Application Type. A 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.

Figure 1 summarizes the Reuse Frame key elements: nodes and refinement relationships. The refinement into more specific and classified aspects requires the specification of a rank for each relationship starting from the same starting node. It is named seq for sequence in Figure 1.

![Fig. 1. Reuse Frame main elements](image-url)
3.2 Reuse Frame Structure

In the Reuse Frame, the root node is mandatory, as well as the 3 main aspects: Human, Organizational and Application domain. Nodes close to the root node are seen as general aspects while nodes close to leaf nodes (including leaf-node) are seen as 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 edge. All starting edges must be from the same kind: if different kinds of edges are required, it means that an intermediary node is needed to clearly group the edges of the same kind. Figure 2 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 whose starting edges are not labeled as exclusive (otherwise, only the ending nodes of the exclusive relationships may be considered as aspects). Figure 2 shows part of a well-formed Reuse Frame. To see the full Reuse Frame, please refer to the Reuse Frame Web Page.

Fig. 2. Part of the Reuse Frame
**Inclusion between aspects** An aspect $a_1$ is included in an aspect $a_2$ if the path from the root node to $a_1$ is a sub-path of the path from the root node to $a_2$. An example is shown in figure 3.

![Figure 3. Example of inclusion between aspects](image)

**Precedence between aspects** An aspect $a_1$ is *previous* an aspect $a_2$ if they have the same upper node and the rank of $a_1$, written $cl(a_1)$, is inferior to the rank of $a_2$, written $cl(a_2)$: $cl(a_1) < cl(a_2)$. It is *next* $a_2$ if they have the same upper node and $cl(a_1) > cl(a_2)$. An example is shown in figure 4.

**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. An example of compatible aspects and an example of non compatible aspects are given in figure 5.

### 3.3 Maintaining the Reuse Frame

Aspects are added to the *Reuse Frame* by MEs. When the addition of node $n_1, ..., n_i$ changes an existing node $n$ into a node with exclusive starting edges, then the *Method Fragment(s)* currently associated to $n$ have to be changed to be associated with one of $n_1, ..., n_i$ aspects ($n$ is not anymore a correct aspect). Aspects cannot be removed except if they do not qualify any *Method Fragment*. *Method Fragments* are associated to or dissociated from aspects by MEs.
3.4 The Reuse Frame in use

The aspects defined in the Reuse Frame aim at supporting capitalization, sharing and reuse based on MR knowledge. In this section, we explain how the Reuse Frame is used to first qualify Method Fragment through the notion of Method
Fragment Reuse Context and then to qualify Reuse Situation through the notion of Method User Reuse Situation. As we also want to retrieve and reuse Method Fragments which do not exactly match the Method User Reuse Situation, we also propose a Similarity Metrics which is also presented in the last part of this section.

**Method Fragment Reuse Context** In SME approaches, Method Fragments are stored in a repository in order to be reused and assembled into an organization-wide situational method. Method fragments need to be qualified by specific slots to be retrieved inside the repository by the MEs during the method building process. MUs need a customized view on the organization-wide situational method, and again method Fragments need to be qualified by specific slots to be then retrieved when matching the MR needs. Retrieval may be supported by keyword matching on specific keyword. But we believe strict matching between keywords is not sufficient. A Method Fragment Reuse Context which does not fully match the keywords is for instance a Method Fragment Reuse Context which keywords are included in the User Situation list of keywords. But with regards to ISD activities, we believe specific kinds of relationships between keywords are meaningful and should be taken into account to retrieve close Method Fragments. MR Knowledge is not something very well defined and each person making reference to it could understand something slightly different about it. In the same way, ISD activities are not very well defined tasks and may slightly differ from one way of working to another. Therefore, guidelines may be more or less detailed in the body of a Method Fragment, and Method Fragment may be qualified by more or less precise keywords. Therefore, we believe it is meaningful, when retrieving Method Fragment to provide means to search also for Method Fragment qualified by more generic or more specific keywords. Looking at knowledge qualifying ISD activities, one may observe that some of them are ordered. In the human perspective, for instance, expert designers know more about design than medium ones, who know more than novice ones. Therefore, a Method Fragment dedicated to an expert designer may also be interesting for a medium one, as well as a Method Fragment dedicated to a novice designer may also be interesting for a medium one. Borderlines between ordered aspects (expert, medium, novice) are not always strictly defined. Therefore, we believe it is meaningful, when retrieving Method Fragments to provide means to search also for Method Fragment associated to keywords previous or next the keywords under consideration in the User Situation. In our Reuse Frame as well as in the retrieval mechanism that we provide, we propose means to deal with more generic and more specific keywords as well as with ordered keywords. To support this retrieval technic, In our approach Method Fragment are qualified through a Method Fragment Reuse Context

The Method Fragment Reuse Context $RC_f$ abbreviated Reuse Context in the following, of a Method Fragment $f$ is defined as the set $C_{RC_f}$ of at least one aspect selected among thus provided in the Reuse Frame. Aspects of $C_{RC_f}$ must be compatible among them and can’t be included one in each other.
\[ RC_f = C_f \] where \( C_f = (a_1, ..., a_n) \) with \( a \in RF \)

Method Fragments providing general guidelines should be qualified with the help of less refined aspects (i.e. corresponding to nodes close to the root node). Method Fragments providing specific guidelines should be 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 the Method Fragment entitled Behavior Investigation is presented in figure 6.

![Behavior Investigation](image)

**Fig. 6. Example of Method Fragment Reuse Context**

**Method User Reuse Situation** To extend the scope of SME by allowing individual team members to adapt their method on the fly, we provide MUs with means to query the Method Fragment repository to find suitable methodological guidelines with regards to their daily jobs. It is supported by the Method User Reuse Situation, abbreviated Reuse Situation in the following. Reuse Situation allows MUs to select aspects among thus stored in the Reuse Frame in order to express the main features of the Method Fragments he/she is interested in.

In the Reuse Situation, in addition to the pertinent aspects, called necessary aspects, MUs may give forbidden aspects, that is to say aspects he/she is 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 between necessary and forbidden aspects: It is not possible by definition to find two incompatible aspects in the same Reuse Context.

The Reuse Situation of a Method User $u$ is defined as a n-uplet $RS_u =<NC_u, FC_u>$ where:

- $NC_u$ is the set of at least one necessary aspect qualifying the MR need of the MU $u$. $NC_u = (a_1, ..., a_i)$ with $a \in RF$
- $FC_u$ is the set of forbidden aspects also qualifying the MR need of the MU $u$. $FC_u = (a_j, ..., a_m)$ with $a \in RF$

If the MU searches for general guidelines, he/she should select necessary aspects among the aspect of the Reuse Frame which are less refined, that is to say aspects corresponding to nodes close to the root node. On the contrary, if the MU searches for specific guidelines, he/she may specify his/her need by selecting in the Reuse Frame aspects among the aspects which are more refined, 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 7.

**Similarity Metrics** By comparing Reuse Context associated to each Method Fragment stored in the repository to the Reuse Situation given by the MU, meaningful Method Fragments may be retrieved from the repository and presented to the MU. This selection may also be done inside the Method Fragment set corresponding to the organization-wide specific method instead of the whole repository. We introduce the Similarity Metrics to quantify the matching between a Reuse Context and a Reuse Situation. This 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:

$$sm(RC_f, RS_u) = \frac{\sum_{i=1}^{n} d(c_{NC_u}, C_f) - \sum_{j=1}^{m} d(c_{FC_u}, C_f)}{\text{card}(NC_u)}$$

where $RS_u$ is the Reuse Situation, $NC_u = c_{NC_{u1}}, ..., c_{NC_{un}}$ its necessary aspect set, $FC_u = c_{FC_{u1}}, ..., c_{FC_{um}}$ its forbidden aspect set, $RC_f$ a Reuse Context, $C_f$ its set of aspects and $d(c, C)$ a closeness distance 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:

$$sm(RC_f, RS_u) = \frac{\text{card}(C_f \cap FC_u)}{\text{card}(NC_u)}$$

Making Similarity Flexible A critical issue in software component reuse is the possibility to answer a query by solutions partially matching the query when searching the repository [30,14]. Recent approaches uses semantic or natural language based means to answer this need and provide answer partially matching the query [34,35]. In our approach, we exploit the relationships stored in the Reuse Frame (refinement, classified refinement).

EXTENDING MATCHING THROUGH NECESSARY ASPECTS: When searching for predefined Method Fragments into the repository, a MU is interested in Method fragments which Reuse Context necessary aspects strictly match his/her Reuse Situation. But Method Fragments including more specific aspects in their Reuse Contexts may also be interesting: 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 Functional Domain Reuse aspect, he/she may also be interested by Method
Fragments matching the Weak Functional Domain Reuse, Medium Functional Domain Reuse and Strong Functional Domain Reuse as shown in figure 8.

MU may also be interested in Method Fragments associated to more general aspects usually providing more general-purpose guidelines which could also be useful. If one still looks for instance at Method Fragments matching the Functional Domain Reuse aspect, he/she may also be interested by the Method Fragments covering the Legacy System aspect (cf Figure 8).

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 aspect.

if one looks for instance at Method Fragments matching the Medium Functional Domain Reuse aspect, he/she may be also interested by the Method Fragments matching the Weak Functional Domain Reuse, and Strong Functional Domain Reuse as shown in figure 9.

EXTENDING MATCHING THROUGH FORBIDDEN ASPECTS: Exploring Reuse Frame refinement relationships 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 Fragments 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 (through classified refinement relationship) means to avoid retrieving Method Fragments whose scope overcomes the aspects given by the MU. One may for instance be interested in Strong Functional Domain Reuse and doesn’t want at all to look at Method Fragments suitable for Medium Functional Domain Reuse and Strong Functional Domain Reuse.

Tuning the selection by allowing or not more general, more specific, previous or next aspects to be included in the necessary and/or forbidden aspects given in the Reuse Situation provides a way for the MU to reduce or enlarge the number of Method Fragments retrieved. If one feel he/she did not find enough Method Fragments with regards to his/her methodological need, he/she may allow more general, more specific, previous and/or next aspects in order to find more Method Fragments. On the contrary, if the set of Method Fragments provided as an answer to his/her need is too large, he/she may enlarge the set of forbidden aspects by allowing more general, more specific, previous and/or next aspects and this
way reduce the number of retrieved Method Fragments. The table presented in figure 10 summarizes the tuning possibilities.

<table>
<thead>
<tr>
<th>Necessary aspects</th>
<th>to search for Method Fragments</th>
<th>to retrieve more Method Fragments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forbidden aspects</td>
<td>to avoid Method Fragments</td>
<td>to retrieve less Method Fragments</td>
</tr>
</tbody>
</table>

![Table of tuning possibilities](image)

**Fig. 10.** Synthesis of the tuning possibilities

When extending the Similarity Metrics between a Reuse Context $RC_f$ and a Reuse Situation $RS_u$ with aspects previous or next the aspects given through the Reuse Situation, as well as aspects more generic or specific than the aspects given in the Reuse Situation, the distance with regards to the reference situation (i.e. aspects given by the MU) has to be quantified to better let the MU understand how much the Method Fragment corresponds to its MR need.

**Quantifying the similarity** Extending the Similarity Metrics may be done with the help of more generic, more specific aspects as well as aspects previous or next the aspects under consideration in the Reuse Situation. The closeness distance between the required aspects (given in the Reuse Situation necessary aspects) and the aspects qualifying the retrieved Method Fragments must be quantified to tell the MU if the Method Fragments strictly match the Reuse Situation or have been retrieved thanks to the extension mechanism. In the following we explain how to compute this closeness distance.

**More generic aspects**: Let $l_{a_{iu}}$ be the number of nodes in the path from the root node to the node corresponding to the aspect $a_{iu}$ under consideration in the Reuse Situation $RS_u$.

Let $l_{a_{jf}}$ be the number of nodes in the path from the root node to the node corresponding to the aspect $a_{jf}$ under consideration in the Reuse Context $RC_f$, $a_{jf}$ belongs to the path from the root node to $a_{iu}$.

The closeness distance between $a_{iu}$ and $a_{jf}$ is:

$$d(a_{jf}, a_{iu}) = 1 - (l_{a_{iu}} - l_{a_{jf}}) / l_{a_{iu}}$$
With regards to the part of Reuse Frame given in figure 11, examples of closeness distance computation with more generic aspects are:

- \( d(a, d) = 1 - (4 - 1)/4 = 0.25 \)
- \( d(b, d) = 1 - (4 - 2)/4 = 0.5 \)
- \( d(c, d) = 1 - (4 - 3)/4 = 0.75 \)

**Fig. 11.** Closeness distance with regards to more generic and specific aspects

**Specific aspects**: Let \( l_{a_{iu}} \) be the number of nodes in the path from the root node to the node corresponding to the aspect \( a_{iu} \) under consideration in the Reuse Situation \( RS_u \). Let \( l_{a_{jf}} \) be the number of nodes in the path from the root node to the node corresponding to the aspect \( a_{jf} \) under consideration in the Reuse Context \( RC_f \).

Let \( l_{max} \) be the number of nodes of the longest path from \( a_{iu} \) to a leaf node \( a_{jf} \). \( a_{iu} \) belongs to the path from the root node to \( a_{jf} \). The closeness distance between \( a_{iu} \) and \( a_{jf} \) is:

\[
 d(a_{jf}, a_{iu}) = 1 - (l_{a_{jf}} - l_{a_{iu}})/(l_{max} - l_{a_{iu}} + 1)
\]

Still with regards to the part of Reuse Frame given in figure 11, examples of closeness distance computation with more specific aspects are:

- \( d(e, d) = 1 - (5 - 4)/3 = 0.66 \)
- \( d(f, d) = 1 - (6 - 4)/3 = 0.33 \)

**Previous aspects**: Let \( a_{iu} \) be the aspect under consideration in the Reuse Situation \( RS_u \) and \( cl_{a_{iu}} \) its classification rank. Let \( a_{jf} \) be the aspect under consideration in the Reuse Context \( RC_f \) and \( cl_{a_{jf}} \) its classification rank. \( a_{iu} \) is next \( a_{jf} \). The closeness distance between \( a_{iu} \) and \( a_{jf} \) is:
\[ d(a_{ji}, a_{iu}) = 1 - (cl_{a_{iu}} - cl_{a_{ji}})/cl_{a_{iu}} \]

With regards to the part of Reuse Frame given in figure 12, examples of closeness distance computation with previous aspect are:

- \[d(1,3) = 1 - (3 - 2)/3 = 0.66\]
- \[d(2,3) = 1 - (3 - 1)/3 = 0.33\]

\[\text{Fig. 12. Closeness distance with regards to previous and next aspects}\]

Next aspects: Let \(a_{iu}\) be the aspect under consideration in the Reuse Situation \(RS_u\) and \(cl_{a_{iu}}\) its classification rank. Let \(a_{ji}\) be the aspect under consideration in the Reuse Context \(RC_f\) and \(cl_{a_{ji}}\) its classification rank. Let \(cl_{max}\) be the highest classification rank among the aspects next \(a_{iu}\). \(a_{iu}\) is previous \(a_{ji}\).

The closeness distance between \(a_{iu}\) and \(a_{ji}\) is:

\[ d(a_{ji}, a_{iu}) = 1 - (cl_{a_{ji}} - cl_{a_{iu}})/(cl_{max} - cl_{a_{iu}} + 1) \]

Still with regards to the part of Reuse Frame given in figure 12, examples of closeness distance computation with next aspects are:

- \[d(6,3) = 1 - (6 - 3)/4 = 0.25\]
- \[d(5,3) = 1 - (5 - 3)/4 = 0.5\]
- \[d(4,3) = 1 - (4 - 3)/4 = 0.75\]

Extension combination: Thanks to the previously defined closeness distance, the extended Similarity Metrics between a Reuse Situation \(RS_u\) and a Reuse Context \(C_f\) becomes the following:
\[ sm(RC_f, RS_u) = \frac{\sum_{j=1}^{n} ed(c_{NC_u}, C_j) - \sum_{j=1}^{m} ed(c_{FC_u}, C_j)}{\text{card}(NC_u)} \]

where \( RS_u \) is the Reuse Situation, \( NC_u = c_{NC_u1}, \ldots, c_{NC_un} \) its necessary aspect set, \( FC_u = c_{FC_u1}, \ldots, c_{FC_un} \) its forbidden aspect set, \( RC_f \) a Reuse Context, \( C_f \) its set of aspects and \( ed(c, C) \) a distance defined as follows:

- \( ENCG_c \) is the set of all the aspects included in aspects from \( NC_{RS_u} \)
- \( ENCS_c \) is the set of all the aspects including aspects from \( NC_{RS_u} \)
- \( ENCP_c \) is the set of all the aspects classified previous aspects from \( NC_{RS_u} \)
- \( ENCN_c \) is the set of all the aspects classified next aspects from \( NC_{RS_u} \)
- \( ENC_c = ENCG_c \cup ENCS_c \cup ENCP_c \cup ENCN_c \), \( ENC_c = c_1^{\text{ENC}_c}, \ldots, c_k^{\text{ENC}_c} \)
- \( EFCG_c \) is the set of all the aspects included in aspects from \( FC_{RS_u} \)
- \( EFCS_c \) is the set of all the aspects including aspects from \( FC_{RS_u} \)
- \( EFCP_c \) is the set of all the aspects classified previous aspects from \( FC_{RS_u} \)
- \( EFCN_c \) is the set of all the aspects classified next aspects from \( FC_{RS_u} \)
- \( EFC_c = EFCG_c \cup EFCS_c \cup EFCP_c \cup EFCN_c \), \( EFC_c = c_1^{\text{EFC}_c}, \ldots, c_p^{\text{EFC}_c} \)
- \( ed(c, X, C) = \max(d(x, C)), x \in EX_c \), in our case with \( X = NC \) and \( X = FC \).

When retrieving Method Fragments from the Reuse Frame, tuning parameters are given in addition to the Reuse Situation, in order to derive the correct \( ENC_{RS_u} \) and \( EFC_{RS_u} \) including more generic, more specific, previous and next aspects only if required. Then, the search is performed eventually using extended sets and matching Method Fragments are presented with their distance to the original Reuse Situation. This search may be conducted by MEs to retrieve Method Fragment from the repository while customizing the method into an organization-specific method. It may also be conducted by MUs to retrieve Method Fragments from the repository or from the organization-specific method while configuring it.

### 4 Concluding Remarks and Future Works

In this paper we presented an approach to help in supporting and personalizing access to methodology for method engineers when customizing methods into organization-specific methods and method users when configuring an organization-specific method to answer their personal method requirement need. This work is part of a complete framework for situational method engineering in which we consider two perspectives of situational method engineering: (1) organization-specific method engineering which aims at satisfying specific information system development project method requirement and (2) method 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 [20]. To achieve this twofold objective (engineering method tailoring and customization), we propose a framework for situational method engineering combining an assembly-based approach for organization-specific method...
construction and a roadmap-driven approach for method user-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 in the second step, it provides support to choose the most adapted path (roadmap) to satisfy the requirements of a particular project engineer within the projectspecific method. The obtained global situational method engineering approach is more rich that 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. The description of the full approach may be found in [20]. We contribute to this framework through our Reuse Frame: a polymorphic and scalable structure to help in better centering the methodological environment on method requirements and allowing to take advantage of situational method engineering and reuse perspectives.

In our approach, we propose an innovative way to customize and configure methods by providing means to qualify Method Fragments and means to select in the repository or in the organization-specific method the Method Fragments which fit method user/method engineer situation. The keystone of our approach is the Reuse Frame which allows to organize Method Fragments in an ontological perspective.

In the future, we would like to improve our approach by extending it to other kinds of stakeholders. We would also like to enrich it by capturing and exploiting experiences, practices and feedback information on the current way of performing method engineer in the organization. It would help in better understanding method requirement and also better supporting method requirement evolution. And finally, we would also like to extend the search mechanisms to information embedded in the Method Fragment body (full text) in addition to the dedicated Reuse Context.


