

LABORATOIRE



INFORMATIQUE, SIGNAUX ET SYSTÈMES  
DE SOPHIA ANTIPOLIS  
UMR 6070

## METHOD CHUNK FEDERATION

*Isabelle MIRBEL*

*Projet EXECO*

Rapport de recherche  
ISRN I3S/RR-2006-04-FR

Février 2006

---

RÉSUMÉ :

L'ingénierie des méthodes a pour but de proposer des solutions efficaces pour construire, améliorer et supporter l'évolution des méthodes de développement.

Dans le domaine de l'ingénierie des méthodes situationnelles, les contributions ont pour but de proposer des techniques ou des outils qui permettent de construire des méthodes adaptées aux spécificités de chaque projet. Mais peu de recherche ont été faites concernant l'adaptation de méthode dans le cadre d'une utilisation au sein d'une organisation un peu plus large, comme une entreprise, dans le but de construire une méthode spécifique et standard pour l'entreprise.

Dans ce contexte, nous proposons une approche qui consiste à fédérer des morceaux de méthode construits à partir des méthodes spécifiques à chaque projet dans le but de permettre à chaque projet de partager ses pratiques méthodologiques sans pour autant imposer à tous une méthode unique et commune.

MOTS CLÉS :

Fragment de méthode, Fédération, Ingénierie des méthodes situationnelles

---

ABSTRACT:

Method Engineering aims at providing effective solutions to build, improve and support evolution of development methodologies.

Contributions, in the fields of situational method engineering, aim at providing techniques and tools allowing to construct project-specific methodologies. But little research has focus on how to tailor such situational methodologies when used as organization-wide standard approaches.

In this context, we propose an approach which consists in federating the method chunks built from the different project-specific methods in order to allow each project to share its best practices with the other projects without imposing to all of them a new and unique organization-wide method.

KEY WORDS :

Method Chunk, Federation, Situational Method Engineering

# Method Chunk Federation

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. Contributions, in the fields of situational method engineering, aim at providing techniques and tools allowing to construct project-specific methodologies. But little research has focus on how to tailor such situational methodologies when used as organization-wide standard approaches. In this context, we propose an approach which consists in federating the method chunks built from the different project-specific methods in order to allow each project to share its best practices with the other projects without imposing to all of them a new and unique organization-wide method.

**Keywords:** Method Chunk, Federation, Situational Method Engineering

## 1 Introduction

Several decades of works have been spent to provide 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 (ISSD). [6,1,5,9].

Experiments show that the provided models and methods have been adapted to each of the different situations in which they have been used. At the end, almost every project has carried out tailoring in order to apply effectively best standard practices. It exists now a lot of variations around a given method, all of them looking suitable for the project it has been customized for but not so easily translatable in a somewhat different project, event inside the same company or organization.

Works in the field of Situational Method Engineering (SME) have proposed solutions to better handle and answer this need for customization at project level [19,2,24,8]. But as it has been emphasized in [3] little research has focus on how to tailor such situational methodologies when used as organization-wide standard approaches.

One solution is to capture and understand all the situational methods used inside each project to build an organization-wide standard method by merging the best practices coming from each of them. This solution requires that method engineers

(i.e. the persons in charge of the methodology in the organization) is able to capture and understand each variation of each method in each project. It is not an easy task, as it is emphasized in [22] : method engineers and method users (i.e. the person applying / using the methodology) lack experience and ability to establish 'home grown' development methodologies or to tailor existing methodologies. It also requires to make each method user accepts and uses the new organization-wide method instead of his/her customized version of his/her method. Again it is not easy because method users prefer lightweight processes/methodologies to heavyweight ones in which they feel more implicated. Lightweight methodologies increase method users involvement on the contrary of heavyweight methodologies where the only significant choices are made by method engineers. Feedback from users shows that methodologies are seen as too prescriptive and too rigid [21].

Therefore we propose an alternative solution which consists in federating the different project-specific methods in order to allow each project to share its best practices with the other projects without imposing to all of them the new organization-wide method.

The remainder of this paper is as follows. In Section 2, we propose a framework for method federation. In Section 3, the *Reuse Frame* and the *Reuse Situation* making possible the federation are presented. Section 4 details and illustrates the *Reuse Situation* and the *Similarity Metrics* we provide to retrieve meaningful method parts and take advantage of the federation. The conclusion and discussion about our future work are proposed in section 5.

## 2 A Framework for Method Chunk Federation

To handle project-specific method federation, support has to be provided to first make the project-specific method federable and then federate the federable best practices.

Making method federable means to break down the method into meaningful atomic parts. It also requires to qualify each part with meaningful keywords in order to make it retrievable by others inside the federation.

The notion of reusable method component has been widely studied in the field of SME. We distinguish proposals about *Method Chunks* or *Method Fragments* and *patterns*. Conceptual patterns for method construction and extension capture generic laws governing the construction of different but similar method [7]. Based on the observation that any method has interrelated aspects, product and process, several authors proposed two types of method components: process fragments and product fragments [10,24,11]. Other authors consider only process aspects and provide process components [9,12], or process fragments [13,15]. In the work of Ralyte, this two aspects are integrated in the same module called *Method Chunk* [25,8,18]. The notion of *Method bloc* proposed by Prakash [23] is similar to the *Method Chunk* as it also combines product and process perspectives into the same modelling component. Both of these notions represent the basic blocks for constructing method from exist-

ing ones. Generally, they are stored in some method repository or method base.

In our work, we started from the notion of *Method Chunk* which seems the most complete and suitable for our purpose. Indeed, in this approach product and process aspects are interrelated, which is necessary to allow reuse and federation. In addition, in this approach, assembly techniques have been widely studied [18] and this is also suitable in our proposal where we help project members (method users and method engineers) to enrich their project-specific method by discovering new guidelines through the federation ; guidelines which could then be assembled / integrated in the project-specific method.

In Ralyte approach, a method is considered as a couple of two interrelated models: product model and process model. The product model of a method defines a set of concepts, relationships between these concepts and constraints for a corresponding schema construction. The process model describes how to construct the corresponding product model. Moreover, a method is viewed as a set of loosely coupled *Method Chunks* expressed at different levels of granularity. A *Method Chunk* is an autonomous and coherent part of a method supporting the realisation of some specific ISSD activities. Such a modular view permits to reuse chunks of a given method in others.

As a part of a method, a *Method Chunk* ensures a tight coupling of some process part of a method process model and its related product part. In the product part, also called product fragment, the product to be delivered by the *Method Chunk* is captured whereas in the process part, also called process fragment, the guidelines allowing to produce the product are given. The interface of the *Method Chunk* captures the reuse context in which the *Method Chunk* can be applied. It is formalised by a couple <situation, intention>, which characterises the situation that is the input of the chunk process and the intention (the goal) that the chunk achieves. Besides, a descriptor is associated to every *Method Chunk*. It extends the contextual view captured in the chunk interface to define the context in which the chunk can be reused. For more details about the structure and content of a *Method Chunk*, please refer to [17].

As it has been highlighted before, making method federable means to provide means to break down methods into reusable autonomous and coherent parts and also to provide means to qualify each method part with meaningful keywords in order to make it retrievable by others. Dedicated efforts have been made, in the field of method engineering, to provide efficient classification and retrieving techniques to store and retrieve *Method Chunks*. Classification and retrieving techniques are currently based on structural relationships among chunks (specialization, composition, alternative, etc.) and reuse intention matching. From our point of view, current classification and retrieving means are not fully suitable for federation of *Method Chunks* because they are supported by the structure of the method they are part of. 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 [27,26]. But

again, domain information does not look like the most suitable information to support federation as projects may belong to different application domains. The only information that will be understandable by every project member (that is to say information which is neither application domain oriented nor project-specific method oriented) and helpful for method engineering is the knowledge about ISSD activities. We believe that knowledge about organizational, technical and human factors is critical knowledge about ISSD activities [4]. Therefore, we propose a *Reuse Frame* aggregating different critical aspects useful to qualify ISSD activities. This polymorphic structure allows to construct a faceted classification of reusable assets dedicated to Method Engineering. The *Reuse Frame* can be seen as an ontology dedicated to ISSD activities. It is shared by all the projects and project members in order to support federation. Indeed, the descriptor associated to every *Method Chunk* which extends the contextual view captured in the chunk interface to define the context in which the chunk can be reused is specified through a set of at least one keywords taken from the *Reuse Frame*. It is called the *Reuse Context* and allows to meaningfully qualify the *Method Chunk* in order to support its reuse through the federation.

Our proposal aims at federating different project-specific methods that is to say allowing each project to share its best practices with the other projects but without imposing to all of them the new organization-wide method. For this purpose, what we want to provide is a mechanism to extract meaningful *Method Chunk* from the federation. A *Method Chunk* may be meaningful because it covers one or several ISSD activities covered by the project-specific method and is therefore an alternative way of working which may be presented to the project member. For this purpose, we defined a *Similarity Metrics* between *Method Chunks* to compare project-specific *Method Chunks* with the whole set of federated *Method Chunks*. A *Method Chunk* may also be meaningful because it covers one or several ISSD activities which are not (well) covered by the project-specific method but the project member searches for guidelines on these activities. In this case also, the *Method Chunk* may be presented to the project member. For this purpose, we defined a *similarity Metrics* between the *User Situation* and a *Method Chunk* to compare project member need with the whole set of federated *Method Chunks*. The *User Situation* is specified through a set of at least one pertinent keywords and a set of forbidden keywords, that is to say aspects of the ISSD he/she is not interested in. All keywords are taken from the *Reuse Frame*.

The main interest of the federation is the ability to propose new *Method Chunks* to project members. Means have to be provided to retrieve as many *Method Chunks* as possible with regards to the project member needs. Therefore *Method Chunks* which *Reuse Contexts* do not fully match the keywords provided by the project member also have to be presented and the similarity between the *User Situation* and the *Reuse Context* has to be quantified. A *Reuse Context* which does not fully match the keywords is for instance a *Reuse Context* which keywords are included in the *User Situation* list of keywords. But with regards to

ISSD activities, we believe specific kinds of relationships between keywords are meaningful and should be taken into account to retrieve *close Method Chunks*. Knowledge about organizational, technical and human factors, which is critical knowledge about information system and software development is not something very well defined and each person making reference to it could understand something slightly different about it. In the same way, ISSD 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 Chunk*, and *Method Chunk* may be qualified by more or less precise keywords with regards to the organizational, technical and human perspectives, even if shared by all the project members. Therefore, we believe it is meaningful, when retrieving *Method Chunk* from the whole set of federated *Method Chunks* to provide means to search also for *Method Chunk* qualified by more generic or more specific keywords. Looking at knowledge qualifying ISSD 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 Chunk* dedicated to an expert designer may also be interesting for a medium one, as well as a *Method Chunk* 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 Chunks* from the whole set of federated *Method Chunks* to provide means to search also for *Method Chunks* 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. Figure 1 shows the framework of our approach.

In the following sections, we will first detail the *Reuse Frame* and the *Reuse Context* making project-specific methods federable. Then, in section 3, we will show how to take advantage of the federation with the help of the *Reuse Situation* and the *Similarity Metrics*.

### 3 Making project-specific method federable

Making method federable means to provide means to break down methods into reusable autonomous and coherent parts and also to provide means to qualify each method part with meaningful keywords. In this section we first present the *Reuse Frame* which aggregates different critical aspects useful to qualify ISSD activities with regards to the organizational, technical and human dimensions [4]. And then, we introduce the *Reuse Context* which allows to meaningfully qualify *Method Chunks* with regards to the *Reuse Frame* in order to support their federation.

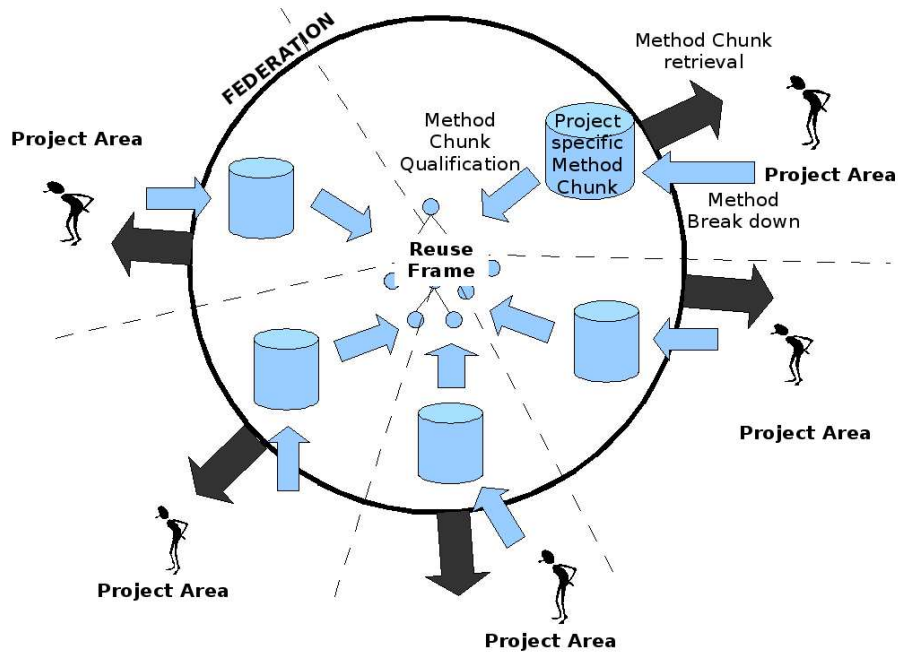


Fig. 1. Method Chunk Federation

### 3.1 The Reuse Frame

In our approach, ISSD knowledge is described in terms of aspects, belonging to aspect families, which are successive refinements of the three main factors of ISSD: human, organizational and technical. Starting from these three basic dimensions, each company may populate the *Reuse Frame* with its own relevant aspects, but we also provide a *Reuse Frame* content that we built from various works made on meaningful aspects for method characterization [17]. With regards to the organizational dimension, we started from the work of van Slooten and Hodes providing elements to characterize ISSD projects [20]: contingency factors, projects characteristics, goals and assumptions as well as system engineering activities. With regards to the Application Domain dimension, we started from previous work on JECKO, a context-driven approach to software development provided in collaboration with the Amadeus Company and proposing a contribution to define software critical aspects in order to get suitable documentation to support software development process [15,16]. The Application Domain dimension also includes aspects related to source system (as legacy system are more and more present in organizations) and application technology, which requires more and more adapted development processes. And finally, about the human dimension, means are provided to qualify the different kinds of method users that may be involved in the ISSD project (analysts, developers, etc.) as well as

their expertise level.

Indeed, the *Reuse Frame* is a tree in which nodes are linked through 3 different kinds of refinement relationships: refinement into node to specify

- more specific aspects,
- more specific and classified aspects,
- more specific and exclusive aspects.

The refinement into node to specify 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 Chunks* to find *Method Chunks* which *Reuse Contexts* include aspects classified *previous* or *next* the aspects of the *Method Chunk* or of the *Reuse Situation* under consideration. The refinement into node to specify exclusive aspects is also another useful kind of relationship. It avoids project members from qualifying *Method Chunks* or *Reuse Situation* through incompatible aspects.

In the *Reuse Frame*, the root node, **base**, is mandatory, as well as the 3 nodes specifying the main aspects: **Human**, **Organizational** and **Application domain**. Nodes close to the root node deal with general aspects while nodes close to leaf nodes (including leaf-node) deal with precise aspects. An *aspect* is fully defined as a path from the root node to a node **n** of the *Reuse Frame*. If **n** is not a leaf node, then it should not have exclusive relationships starting from it, otherwise one of the ending node of the exclusive relationships has to be chosen as **n**. *Inclusion* between aspects has been defined to specify when an aspect is more generic or more specific than another one. The *Precedence* relationship has also been defined to specify when an aspect is *previous* or *next* another one. And finally *compatibility* between aspects to allow them to be part of the same *Reuse Situation* or *Reuse Context* has also been defined. Figure 2 shows part of a well-formed *Reuse Frame*. In this part of the *Reuse Frame*, **Source System** is an example of *wrong* aspect while **Legacy System** is an example of *right* one; **Legacy System** is included in **Functional Domain Reuse**; **Strong Reuse** is an aspect more specific than **Code Reuse** and **Legacy System** is more generic than **Code Reuse**; **Strong Reuse** is *next* **Medium Reuse** while **Weak Reuse** is *previous* **Medium Reuse**; and finally, **No Source System** and **Functional Domain Reuse** are not compatible aspects while **Virtual User** and **Real User** are compatible. For the full description of the rules to build a well-formed *Reuse Frame* please refer to [14].

### 3.2 The Reuse Context

The *Reuse Context* is defined as a set of at least one compatible aspect taken from the *Reuse Frame*. *Method chunks* providing general guidelines are usually associated to general aspects, that is to say aspects which paths is ended by nodes close to the root node. On the contrary, specific guidelines are provided in *Method Chunks* associated to precise aspects, that is to say aspects which paths

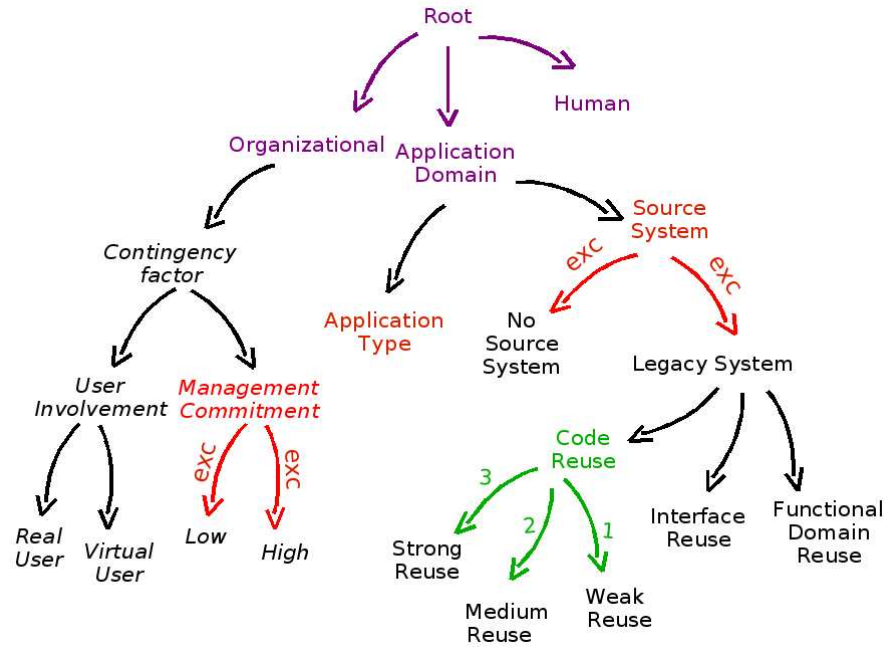


Fig. 2. The Reuse Frame - An example

is ended by nodes close to leaf-nodes or leaf-nodes themselves. It is up to the project member who enters the *Method Chunk* into the project-specific method chunk repository to select the most meaningful aspects to qualify the *Method Chunk*.

## 4 Supporting Method Chunk Federation

In this section we present the *User Situation* allowing project members to specify their need and the *Similarity Metrics* to compare project member need or project-specific method chunk with the whole set of federated *Method Chunks*. Then, we detail how we can extend our *Similarity Metrics* to retrieve *Method Chunks* which *Reuse Contexts* including more generic or more specific aspects, as well as aspects classified as previous or next the aspects of the *Method Chunk* under consideration.

### 4.1 User Situation

The *Reuse Situation* allows project members to select aspects among thus stored in the *Reuse Frame* in order to express the main features of the *Method Chunks* he/she is interested in. In the *Reuse Situation*, in addition to the pertinent aspects, called *necessary aspects*, project members 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 Chunks* including these (forbidden) aspects will not appear in the retrieved set of *Method Chunks* answering the methodological need. All aspects must be compatible among each others 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*

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

## 4.2 Similarity Metrics

The main goal of our work is to provide means to federate different project-specific methods in order to allow each project member to share the best practices of his/her project with the members of the other projects without imposing to all of them a new and unique organization-wide method built from all the project-specific methods. Therefore, we provide a mechanism to retrieve meaningful *Method Chunks* from the federation. A *Method Chunk* may be meaningful because it covers one or several ISSD activities covered by the project-specific method and is therefore an alternative way of working which may be presented to the project member. The *Reuse Context* of the two *Method Chunks* under consideration (one project-specific *Method Chunk* and one *Method Chunk* from the federation) have to be compared to check if the *Method Chunk* from the federation covers the ISSD activities covered by the project-specific *Method Chunk*. By comparing the number of common aspects in their *Reuse Contexts*, a *Similarity Metrics*, varying between 0 and 1, is computed to indicate to the project member how much the *Method Chunk* from the federation matches the project-specific *Method Chunk*.

A *Method Chunk* may also be meaningful because it covers one or several ISSD activities which are not (well) covered by the project-specific method but the project member searches for guidelines on these activities. In this case, the retrieval is done by comparing the *Reuse Context* of the *Method Chunk* from the federation with the *Reuse Situation* specifying the project member need.

In this case, 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*. A positive value of the *Similarity Metrics* indicates that there are more necessary aspects than forbidden ones in the *Reuse Context* with regards to the *Reuse Situation*. On the contrary, a negative value indicates that there are less necessary aspects than forbidden ones. The perfect adequation is represented by the value 1.

### 4.3 Extended Similarity Metrics

When searching for *Method Chunks* inside the federation, *Method Chunks* 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 project member is interested in. Project member may also be interested in *Method Chunks* associated to more general aspects usually providing more general-purpose guidelines which could also be useful. In the same way, the classification dimension of refinement relationships may be exploited to enlarge the set of *Method Chunk* retrieved with *Method Chunks* which *Reuse Contexts* include *previous* or *next* aspects.

Exploiting *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 Chunks* associated to too specific aspects, most probably qualifying *Method Chunks* 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 Chunks* whose scope overcomes the aspects given by the project member.

Extending 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 project member to reduce or enlarge the number of *Method Chunks* retrieved. If one feel he/she did not find enough *Method Chunks* 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 Chunks*. On the contrary, if the set of *Method Chunks* 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 Chunks*. The table presented in figure 3 summarizes the extension possibilities.

When the *Similarity Metrics* is computed with extended necessary and forbidden aspects, a distance has to be provided to quantify the closeness between the aspect under study and the more generic, more specific, previous or next aspects. Therefore, we propose 4 distances to quantify the closeness between two aspects.

	Exact Matching	Extended Matching		
		Less refined aspects	More refined aspects	before/after aspects
Necessary aspects	to search for Method Chunks	to retrieve <i>more</i> Method Chunks		
		More general Chunks	More specific Chunks	Adjacent Method Chunks
Forbidden aspects	to avoid Method Chunks	to retrieve <i>less</i> Method Fragments		
		to avoid full branches of the Reuse Frame	to avoid too specific Chunks	to avoid adjacent / overlapping Chunks

**Fig. 3.** Similarity Metrics - Extension possibilities

- The *Closeness distance* between an aspect **a1** and another 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.
- The *Closeness distance* between an aspect **a1** and another 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 a leaf-node.
- The *Closeness distance* between an aspect **a1** and another 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**.
- The *Closeness distance* between an aspect **a1** and another 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**.

A perfect matching between the 2 aspects leads to the value 1 of the *Closeness distance*, which tends to 0 as far as the ratio decreases. Figure 4 shows the different situations of *Closeness Distance* computation. Examples of aspects *more generic*, *more specific*, *previous* and *next* may be found in figure 2.

The *similarity Metrics* is expendable thanks to this *Closeness distance*. When the aspect which is present in the *Reuse Situation* of the *Method Chunk* 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 Metric*. For the full description of the *Closeness Distance* and the *Similarity Measure*, please refer to [14].

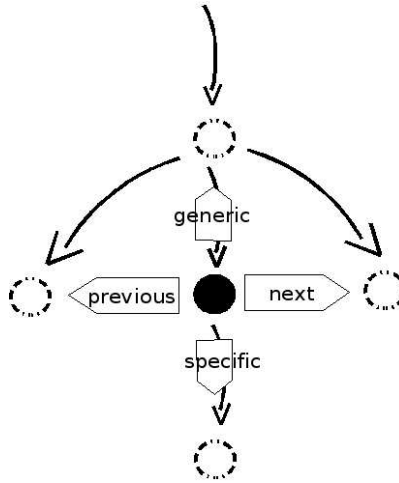


Fig. 4. Closeness Distance computation

## 5 Concluding Remarks and Future Works

In this paper we presented an approach to support federation of project-specific method in order to allow each project to share its best practices with the other projects without imposing to all of them the new organization-wide method. We started from the work of Ralyte about *Method Chunk* to break down project-specific methods into atomic and reusable parts [18]. Our contribution focusses on the specification and use of a *Reuse Frame* to retrieve meaningful *Method Chunks*. For this purpose, we provide means to:

- Make the federation possible by introducing the *Reuse Frame* in order to capture and share knowledge about ISSD activities and the *Reuse Context* to allow project members to qualify the content of each atomic and reusable part of the project-specific method.
- Support de federation by providing means for the project member to express his/her need through a *Reuse Situation* and by proposing a *Similarity Metrics* and a *Closeness Distance* to retrieve *Method Chunk* not strictly matching the *Reuse Situation* by exploiting the genericity and classification relationships which exist in the knowledge qualifying ISSD activities.

In the future, we would like first to try our approach on a real case study and we would also like to improve it by enriching the *Reuse Frame* with a view or tag mechanism allowing each project or each project member to associate its own vocabulary to the aspects defined in the *Reuse Frame*, and this way better exploit ISSD knowledge.

## 6 References

- [1] A. Rochfeld – **MERISE, an Information System Design and Development Methodology** – ER, 1986.
- [2] A.F. Harmsen – **Situational Method Engineering** – *Moret Ernst Young*, 1997.
- [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] C. Cauvet and C. Rosenthal-Sabroux – **Ingenierie des systemes d’information** – *Hermes*, 2001.
- [5] C. Rolland, C. Cauvet – **Object Oriented Conceptual Modelling** – CISM0D’92 International Conf. on Management of Data, Bangalore, July, 1992.
- [6] C. Rolland, O. Foucault, G. Benci – **Conception des systèmes d’information: la méthode REMORA** – *Eyrolles*, 1988.
- [7] C. Rolland, V. Plihon – **Using Generic Method Chunks to Generate Process Models Fragments** – International Conference on Requirement Engineering, 1996, pp. 169-194.
- [8] C. Rolland, V. Plihon, J. Ralyté – **Specifying the Reuse Context of Scenario Method Chunks** – International Conference on Advanced Information System Engineering, 1998.
- [9] D.G. Firesmith, B. Henderson-Sellers – **The OPEN Process Framework - An Introduction** – *Addison-Wesley*, 2002.
- [10] 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.
- [11] H.T. Punter, K. Lemmen – **The MEMA model: Towards a new approach for Method Engineering** – *Information and Software Technology*, 38(4), 1996, pp. 295-305.
- [12] I. Graham, B. Henderson-Sellers, H. Younessi – **The OPEN Process Specification** – *Addison-Wesley*, 1997.
- [13] I. Mirbel – **Rethinking ISD methods: fitting project team members profiles** – ISD 2004, Vilnius, Lituania, September, 2004.
- [14] I. Mirbel – **Method Engineering: A user-centric contribution** – I3S/RR-2006, I3S Laboratory, 2006.
- [15] 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.
- [16] I. Mirbel and V. de Rivieres – **Conciliating User Interface and Business Domain Analysis and Design** – OOIS 2003, Geneva, Switzerland, September, 2003, pp. 383-399..
- [17] I. Mirbel, J. Ralyte – **Situational method engineering: combining assembly-based and roadmap-driven approaches** – *Requirement Engineering Journal*, 11(1), 2006, pp. 58-78.
- [18] J. Ralyte – **Ingenierie des methodes a base de composants** – Universite Paris I - Sorbonne, January, 2001.
- [19] K. Kumar, R.J. Welke – **System Analysis and Design: a research agenda** – *wiley*, 1992.
- [20] K. van Slooten and B. Hodes – **Characterizing IS Development Projects** – IFIP TC8, WG 8.1/8.2, August, 1996, pp. 29-44.

- [21] M. Bajec, D. Vavpotic, M. Kirsper – **The scenario and tool-support for constructing flexible, people-focused system development methodologies** – ISD 2004, Vilnius, Lithuania, September, 2004.
- [22] 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.
- [23] N. Prakash – **On Method Statics and Dynamics** – Information Systems, 34(8), 1999, pp. 613-637.
- [24] S. Brinkkemper, M. Saeki, F. Harmsen – **Assembly Techniques for Method Engineering** – International Conference on Advanced Information Systems Engineering, 1998.
- [25] V. Plihon, J. Ralyté, A. Benjamen, N.A.M. Maiden, A. Sutcliffe, E. Dubois, P. Heymans – **A Reuse-Oriented Approach for the Construction of Scenario Based Methods** – International Conference on Software Process, 1998.
- [26] V. Pujalte, P. Ramadour – **Réutilisation de composants: un processus interactif de recherche** – Majestic'05, 2004.
- [27] V. Sugumaran, V.C. Storey – **A semantic-based approach to component retrieval** – The database for advances in information systems, 34(3), 2003.