Requirements Definition for the Situational Method Engineering

Jolita Ralyté
CUI, Université de Genève, 24 rue du Général Dufour, CH-1211 Genève 4, Switzerland
ralyte@cui.unige.ch

Abstract: The work presented in this paper is related to the situational method engineering domain, which focus on the project-specific method construction. The techniques proposed by this discipline aim to construct methods by assembling reusable method fragments. The selection of the method fragments is requirements driven. The first step in the situational method engineering consists in analysing the situation of the current project and specifying the requirements for a method supporting this situation. In this paper we propose an approach supporting the definition of the requirements for a situational method. The approach considers different method engineering situations and provides guidelines supporting method requirements definition in each of them. The application examples illustrate our approach throughout the paper.

1. INTRODUCTION

The increasing complexity of the Information Systems (IS) asks for new IS engineering methods, which could take into account the specific situation of each IS development project and help to manage thus its complexity. However, the traditional method construction techniques are too expensive and time-consuming and are not suitable for a project-specific method construction. The Situational Method Engineering (SME) discipline emerges as a reaction to this problematic. Brinkkemper defines the SME as *"the discipline to build*

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35614-3_21
C. Rolland et al. (eds.), *Engineering Information Systems in the Internet Context*

project-specific methods, called situational methods, from parts of the existing methods, called methods fragments" [2]. Hence, the objective of the SME is to provide rapid method engineering techniques based on the reuse of the existing method parts in the construction of new methods, more flexible and better adaptable to the situation of every IS development project.

Different approaches have been proposed to achieve this goal. These approaches introduce the notions of *method fragment* [6, 18] and *method chunk* [9, 15, 19] as the basic blocks for constructing 'on the fly' methods. Some authors propose repositories for method fragments/chunks storage [15, 19, 23]. The process model for reengineering of the existing methods into collections of method chunks is proposed in [13]. This model helps to identify the reusable parts of the existing methods, which are not modular in origin, and to specify them as method chunks, which can be stored in a method base. In addition, there are a number of proposals for approaches to assemble these fragments/chunks [3, 9, 11, 12, 14, 24]. Following these approaches, new methods can be constructed by selecting the fragments/chunks that are the most appropriate to a given situation from the method repository. As a result, SME favours the construction of modular methods that can be modified and augmented to meet the requirements of a given situation [6, 26].

The work presented in this paper is related to the SME domain and focus on the method requirements definition technique. To enable 'on the fly' method construction we need to define first the requirements for a new method. These requirements are used next by the method chunk selection and assembly process that we have presented in [12].

This paper is organised as follows: section 2 provides an overview of our SME approach. Sections 3 and 4 describe and illustrate two method requirements elicitation processes. Section 5 draws some conclusions and discussions around our future work.

2. OVERVIEW OF THE SITUATIONAL METHOD ENGINEERING PROCESS

The objective of the SME is to adapt existing methods or to construct new ones according to the situation of the current IS development project. As a consequence, the first step in the SME process consists

in analysing the situation of the current project and specifying the requirements for a method supporting this situation. Afterwards, the second step consists in constructing the method following these requirements.

The method construction depends of the objective of the SME. There are many different reasons for constructing a new method. We identified four of them:

1. To define a brand new method to satisfy a set of situational requirements;
2. To add alternative ways-of-working in a method to its original one;
3. To extend a method by a new functionality;
4. To select in a method only relevant functionalities.

Each of these delineates a specific strategy for method engineering that we have embedded in our Method Engineering Process Model (MEPM) [12]. The first strategy, called *From scratch,* is pertinent in situations where either there is no method in use or the used one is irrelevant for the project (or class of projects) at hand. The second one, *Completeness driven assembly strategy*, is relevant when the method in use is strong from the product point of view but weak from the process viewpoint. Enhancing the process model of the existing method by one or several new ways of working is thus the key motivation for method engineering. The third strategy, named *Extension driven assembly strategy,* is required in situations where the project at hand implies to add a new functionality to the existing method, which is relevant in its other aspects. Finally, the fourth strategy is appropriated in situations where the project at hand does not need to use all the functionalities proposed by the usually applied method. This strategy, named *Restriction driven strategy*, helps to select the functionalities, which are significant in the project situation and to eliminate the others. These four method construction strategies demonstrate that requirements elicitation process depends of the initial method situation in the project at hand. There are two possible situations:

1. The IS development crew is in the habit to use the same method for all IS development projects but consider that in some situations this method must be adapted.
2. The IS development crew does not posses any regular method.

Each of these situations characterizes a specific strategy for method requirements elicitation and is included in our MEPM [12]. The first

strategy is based on the analysis of the existing method and the detection of the engineering activities (engineering intentions), which must be included into this method, eliminated from it or the achievement of which must be enhanced by new ways of working. We call this strategy – *Intention driven strategy*. The second strategy is based on the identification of the engineering activities, which must be supported by the new method. The identified activities are organised in the manner to form the process model of the required method and the strategy is called – *Process driven strategy*.

This way of thinking shows us that we have several different manners to achieve the method engineering objective that is to construct a new method or to adapt an existing one. For this reason, it seems for us that the more suitable process model to represent our MEPM is the strategic process meta-model proposed in [1, 22] and called a *map*.

The two fundamental concepts in the map are *intention* and *strategy*. An intention **I** is a goal that can be achieved by the performance of the process. It refers to a task (activity) that is a part of the process and is expressed in the intentional level. The strategy **S** represents the manner in which the intention can be achieved. The *map* is a directed labelled graph with nodes representing intentions and labelled edges expressing strategies. The directed nature of the map identifies which intention can be done after a given one. A map includes two specific intentions, *Start* and *Stop*, to begin and end the process respectively. There are several paths from *Start* to *Stop* in the map for the reason that several different strategies can be proposed to achieve the intentions. A map therefore includes several process models that are selected dynamically when the process proceeds, depending on the current situation.

The core notion of the map is a *section* defined as a triplet <*source intention, target intention, strategy*> $<I_i, I_j, S_{ij}>$. Each section is defined by an *intention achievement guideline (IAG)*, which provides advice to fulfil the target intention I_j following the strategy S_{ij} given the source intention I_i has been achieved. Furthermore, a section can be refined as an entire map at a lower level of granularity.

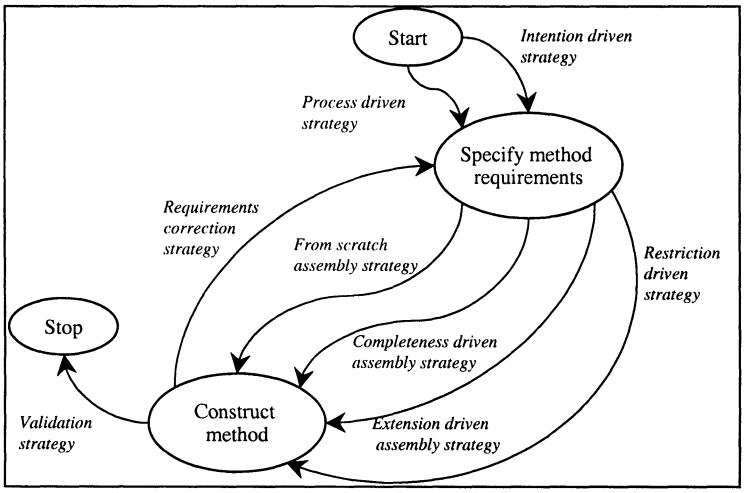

Figure 1. The method engineering process model MEPM.

Our MEPM map is shown in Figure 1. It includes two main intentions *Specify method requirements* and *Construct method.* The latter corresponds to the method engineering essential goal, whereas the former is the prerequisite for the latter. The formulation of this intention, *Specify method requirements* means that our approach is requirements-driven. As stated above, in order to construct a new method, we propose to start by eliciting requirements for the method engineering activity. This is possible following one of two strategies (*Intention driven* and *Process driven)* presented above. Both lead to a set of requirements expressed as a map that we call the *requirements map.* However, each strategy corresponds to a different way of eliciting the requirements. The former is based on the inventory of engineering goals whereas the latter infers these goals from an analysis of the engineering activities that must be supported by the method.

Once the requirements have been elicited, the intention *Construct method* can be achieved. The four assembly strategies (*From scratch, Completeness driven, Extension driven* and *Restriction driven*) proposed by the MEPM correspond to the four method engineering situations that were identified and motivated above. Backtracking to the requirements definition is possible thanks to *the Requirements correction strategy.* Finally, the *Validation strategy* helps verifying

that the assembly of the selected method chunks satisfies all requirements and ends the method engineering process.
In the previous paper [12] we have presented and illustrated some guidelines associated to the MEPM sections dealing with the method construction that is, the assembly of method chunks. This paper focuses on the refinement of the guidelines supporting the specification of the requirements for the situational method that are, the *Intention driven* and the *Process driven* strategies of the MEPM map (Figure 1).

3. PROCESS DRIVEN STRATEGY FOR REQUIREMENTS SPECIFICATION

As mentioned in the previous section, the *Process driven strategy* for method requirements specification is relevant in the case when we need to construct a completely new method. Following this strategy, the requirement elicitation process consists in identifying requirements for a new method in terms of the general intentions to satisfy in the IS development process and the strategies to satisfy these intentions. Consequently, the requirements for a new method must be expressed in the form of a map that we call the *requirements map*. We adopt the multi-process approach of [1] for requirements map construction and the linguistic-based goal structure proposed in [10] for intention formulation.
The meta-process proposed in [1] provides the guidelines for map construction specified in form of contexts. The two principal steps in this approach are (1) the definition of the map sections and (2) the definition of the associated guidelines. Our requirements map construction is limited to the definition of its sections. The definition of the associated guidelines is a part of the method chunk selection and assembly process.
A map in Figure 2 expresses our process driven strategy for method requirements specification as a requirements map. According to this process model, we start the requirements map construction following the *Activity driven strategy*. The guideline associated to this map section helps to define the principal activities of the required method and to formalise them as sections of the requirements map.

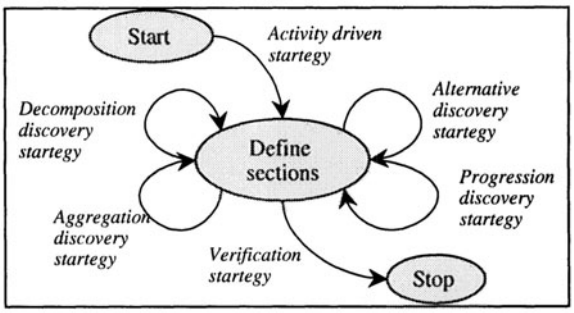

Figure 2. The process driven requirements elicitation model.

Analysis of the already defined sections may imply the definition of new sections or the modification of the existing ones. As shown in Figure 2, the obtained requirements map can be refined and other sections can be discovered from the existing ones following four strategies:

- The *Decomposition discovery strategy* helps decomposing an existing section in several ones.
- The *Aggregation discovery strategy* advises how to combine a set of sections into a new one.
- The *Alternative discovery strategy* helps to identify a new section having an alternative strategy or an alternative source or target intention to the existing one.
- The *Progression discovery strategy* helps to define a new section allowing to progress in the method map from the existing one.

Finally, the *Verification strategy* allows to finish the requirements elicitation process after the documentation of the obtained map sections and the verification of its coherence and completeness.

We detail next the guidelines associated to the sections of this process model. The notion of a context, defined by a couple <*situation, intention*>, is used to specify these guidelines. The *intention* expresses the objective to attain in the requirements elicitation process whereas the situation defines the products necessary to achieve the corresponding intention.

3.1 Activity Driven Strategy

The guideline supporting the *Activity driven strategy* is formalised as follows:

G1: <Project description; Define sections of the requirements map following activity driven strategy>
G1.1: <Project description; Define the signature of the requirements map>
G1.2: <The map signature is defined; Identify the core and essential intentions necessary to achieve the intention specified in the map signature>
G1.3: <Intentions are elicited; Abstract intentions>
G1.4: <Intentions are elicited; Discover strategies>
G1.5: <Intentions and strategies are defined; Define sections>

This guideline proposes to define first the signature of the required method map expressing the global objective of the method (G1.1) and then to identify the core activities necessary to achieve the intention specified in the map signature (G1.2). We use the goal structure proposed in [10] for intention formulation. A goal is expressed as a natural language statement comprising a verb and several parameters like object, result, source, destination, manner etc., where each parameter plays a different role with respect to the verb. This facilitates to explicitly specify the semantics of different parts of a goal. Next, the guideline proposes to make sure that all elicited intentions are at the same level of abstraction (G1.3). The following rules must be verified:

R1: There is not an intention that can be considered as a subset of another one.

R2: There is not an intention that appears to be a way to achieve another one.

Once the intentions have been elicited, we need to discover the possible strategies (G1.4) to achieve these intentions. In other words, we need to consider different manners to reach every intention with respect that strategies, like intentions, also must be at the same level of abstraction and verify the following rules:

R3: There is not a strategy that appears to be a part of one another.

R4: There is not a strategy that specifies the way to an intention at the implementation level.

After that, the map sections can be defined as follows (G1.5):

For every intention I and one of its strategies S, identify $(I,S)_{pre}$ and $(I,S)_{post}$.
Connect sections using the following section construction predicate:

Map: Intention, Intention, Strategy → Section I_i, I_j, S_{ij} → Section (I_i, I_j, S_{ij}) if ($\exists$ $S_{ki} \in$ Strategy: $(I_i, S_{ki})_{Post} \Rightarrow (I_j, S_{ij})_{Pre}$ AND $(I_j, S_{ij})_{Pre} \Rightarrow (I_j, S_{ij})_{Post}$

That means, for every intention I and one of its strategies S we need to identify the situation in which this strategy can be applied (its precondition $(I,S)_{pre}$) as well as the result of the intention achievement following this strategy (its post-condition $(I,S)_{post}$). The precondition $(I,S)_{pre}$ corresponds to the product necessary for the execution of the strategy S in order to achieve the intention I and can be determined by identifying the intention, which produces this product. The post-condition $(I,S)_{post}$ corresponds to the product obtained by executing the intention I following the strategy S. The obtained sections must be documented in the informal manner in order to facilitate the retrieval of the appropriate method chunks.

To illustrate our approach we propose an exemple of the IS project that we have developed for one Swiss watchmakers company. The objective of this project was to develop a B2B system intended to support the information exchange between the company and its subsidiaries and agents distributed in the world. The system ought to support the principal task of the subsidiaries and agents, which is the construction of their purchase plans for the next year as well as the responsibilities of the market service of the company, which consist in validating and consolidating purchase plans and providing the information about delivery terms. A particular method supporting analysis and design of such system is indispensable to take into account the high interactivity level of the system. The signature of the corresponding requirements map can be defined as:

<B2B project described, Construct a method supporting analysis and design of a B2B system>

It is clear, that the step of functional requirements elicitation and conceptualisation is very important for the realisation of such system. Besides object structure for data storage, dynamic perspective of the system is also very important. It must specify life cycles of principal objects as well as global dynamic view representing external, internal and temporal events arriving in the system and how the system reacts to them. The hyperlink navigational structure, typical to web based systems, must also be taken into account. As a consequence, we identify the following intentions:

- *Discover requirements*
- *Validate requirements*
- *Construct structural view*
- *Construct object behavioural view*
- *Construct global behavioural view*
- *Construct navigation structure*

Considering the abstraction level of these intentions we may notice that the intention *Validate requirements* is at the lower level of detail in comparison with other intentions and can be seen as a part of the intention *Discover requirements.*

Next, we consider the different strategies to achieve these intentions. For exemple, to discover functional system requirements we prefer to use a *Goal/scenario-based* approach. Requirements validation step may be seen as a strategy in the requirements discovery process. For the structural view of the system we prefer to have two possibilities: *Preliminary object modelling* approach and *Detailed object modelling* approach. A *State-based modelling* approach could be used for object behavioural view construction whereas the global system behavioural view could be supported by an *Event-based modelling* approach. Finally, we need to connect the intentions and strategies to obtain a requirements map. It is clear that the first step in this process is the requirements discovery step. The creation of all other models depends of the obtained requirements specification. The precondition for the construction of the object behavioural view is the existence of the structural view defining the participating objects. Global behavioural views requires as precondition the life-cycle models of different objects that is the object behavioural view. Lastly, the construction of the web site navigation structure requires as a precondition the existence of the global behavioural view of the system. Figure 3 illustrates the obtained requirements map.

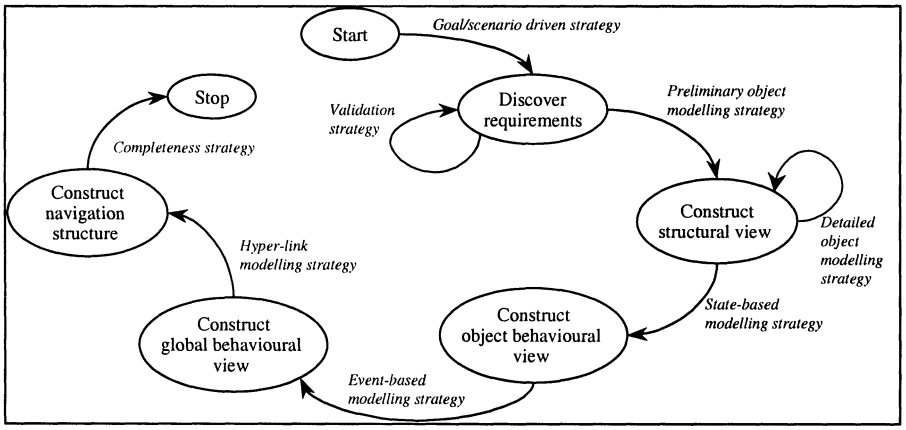

Figure 3. The initial requirements map.

3.2 Aggregation Discovery Strategy

The guideline associated to the *Aggregation discovery strategy* helps to group together some sections by aggregating their target intentions or by aggregating their strategies. The intentions can be aggregated if one of the following conditions is truth:

C1. There exist two or more intentions resulting in the same kind of products.

C2. There exist two or more intentions mutually complement and usually going together.

C3. There exist two or more intentions belonging to the consecutive sections and there exists only one strategy for each intention.

The section aggregation is based on the grouping of the strategies of parallel sections and is controlled by the condition C4. Parallel sections are those that have the same source and target intentions but different strategies.

C4. There exist several parallel sections and their strategies represent different tactics of the same manner to achieve the target intention.

Therefore, the guideline is specified as follows:

```
G3: <A section is defined; Define sections following
aggregation discovery strategy>
    G3.1: If C1 or C2 or C3 is truth then <The intentions
are identified, Aggregate intentions>
    G3.2: If C4 is truth then <The parallel sections are
identified, Aggregate strategies>
```

In the requirements map shown in Figure 3, the intentions *Construct object behavioural view* and *Construct global behavioural view* could be aggregated for the reason that both views are closely related and represent dynamic perspective of the system. We propose to aggregate these two intentions into a new one that we call *Construct dynamic view*. The obtained requirements map is shown in Figure 4.

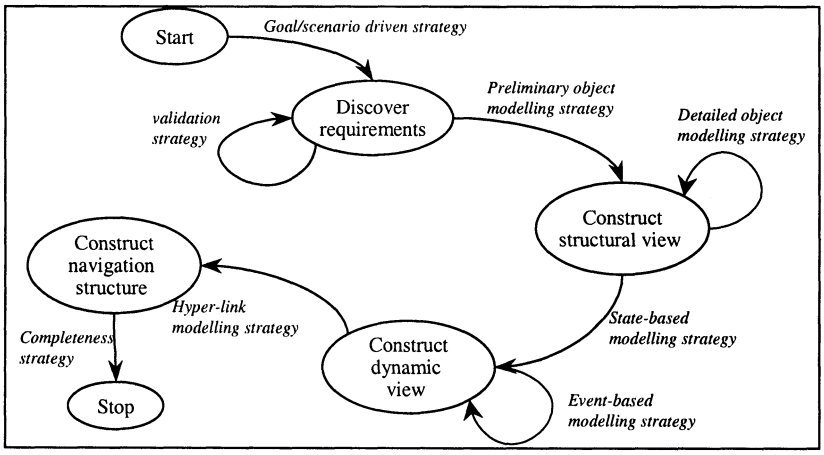

Figure 4. The requirements map after application of the aggregation discovery strategy.

3.3 Decomposition Discovery Strategy

The guideline supporting the *Decomposition discovery strategy* permits to split some sections into several ones in two alternative ways: (1) by decomposing an intention if the condition C5 is satisfied or (2) by decomposing a strategy if the condition C6 is satisfied.

C5: There exists an intention with the level of granularity higher than this of the other intentions.

C6: There exists a strategy containing several different manners to achieve the target intention.

In the first case we need to discover strategies for every new intention and to define the new sections. In the second case we obtain a set of parallel sections. This guideline is specified as follows:

```
G2: <A section is defined; Define sections following
decomposition discovery strategy>
    G2.1: If C5 is truth then <An intention is
identified, Decompose intention>
    <New intentions are defined, Discover strategies>
    <Intentions and strategies are defined; Define
sections>
    G2.2: If C6 is truth then <A strategy is identified,
Decompose strategy>
```

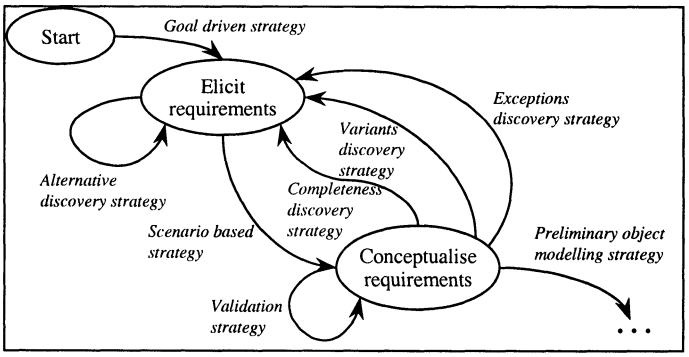

Figure 5. The requirements map after application of the decomposition strategy.

In our example (Figure 4), we decide to decompose the section <*Start, Discover requirements, Goal/scenario-based strategy*>. Its target intention, *Discover requirements*, represents a complex process and could be decomposed into two sub-intentions: *Elicit requirements* and *Conceptualise requirements*. Different strategies can be identified to realise these sub-intentions. For exemple, *Alternative discovery strategy* could be added in order to elicit requirements representing alternative functional solutions to the existing ones; *Variant* and *Exception* discovery strategies are necessary to guaranty requirements completeness and can be applied on the already conceptualised requirements. New sections are constructed by using the two identified sub-intentions and the above-mentioned strategies. The concerned fragment of the resulting map is shown in Figure 5.

3.4 Alternative Discovery Strategy

The *Alternative discovery strategy* helps to discover new sections by considering other possible ways to achieve the same target intention or by identifying alternative intentions. In the first case the condition C7 is verified whereas in the second case the condition C8 must be satisfied.

C7: There exist alternative strategies to achieve an intention.

C8: There exist alternative intentions to a target intention.

```
G4: <A section is defined; Define sections following
alternative discovery strategy>
    G4.1: If C7 is truth then <An intention and a
strategy are defined; Define section>
    G4.2: If C8 is truth then <An intention is defined;
Discover strategies>
    <Intention and strategies are defined; Define
sections>
```

In our example, we consider other ways to reach intention *Elicit requirements*. Besides *Goal driven strategy*, we propose to add *Actor-based discovery strategy*. As a result, our requirements map is enriched by a new section *<Start, Elicit requirements, Actor-based discovery strategy>*. The refined map fragment is illustrated in Figure 6.

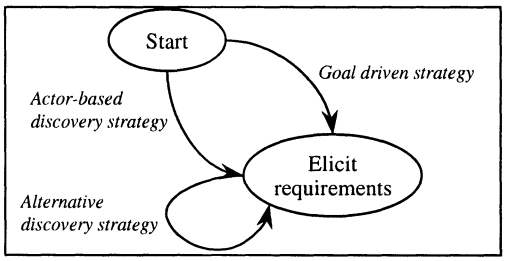

Figure 6. The requirements map after application of the alternative discovery strategy.

3.5 Progression Discovery Strategy

The *Progression discovery strategy* helps to discover other sections, which follow an existing one. In other words, it helps to find new

sections, which allows to progress from the situation created by a given section to another one. This guideline is defined as follows:

```
G5: <A section is defined; Define sections following
progression discovery strategy>
    G5.1: <A section is selected; Identify a target
product>
    G5.2: <A target product is identified; Define an
intention requiring this product as source>
    G5.3: <An intention is defined; Discover strategies>
    G5.4: <Intention and strategies are defined; Define
sections>
```

To illustrate this guideline, we consider the section *<Construct structural view, Construct dynamic view, State-based strategy>* of our requirements map (Figure 4). This section supports the construction of a model representing object life cycle. This kind of models generally uses the notion of activity (or operation) responsible of the object state transition. Some activities can be complex enough and require detailed specification. As a consequence, we need to add new intention *Construct activity view*. However, this intention is a part of the intention *Construct dynamic view*. Therefore, the new section to add is a recursive one *<Construct dynamic view, Construct dynamic view, Activity-based modelling strategy>*. The final requirements map is illustrated in Figure 7.

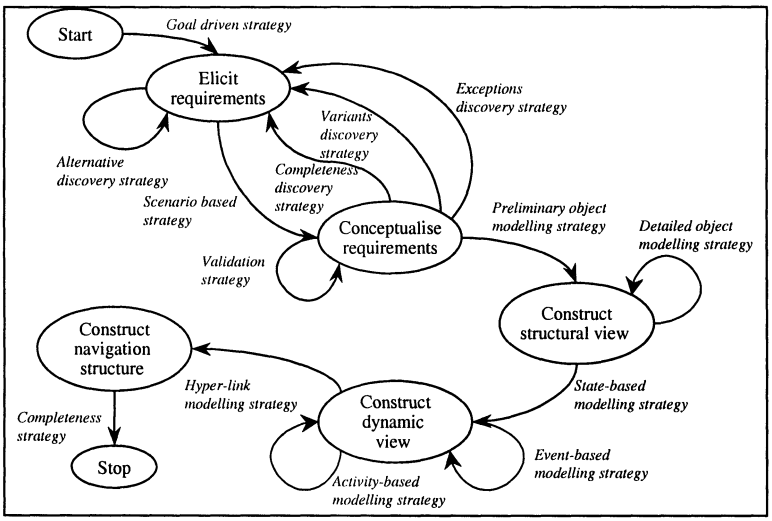

Figure 7. The final requirements map.

3.6 Role of the Requirements Map in the Assembly Based Method Construction Process

As mentioned in the section 2 of this paper, the requirements map is used as a basis for the method chunk selection and assembly process. This process is based on the matching mechanism between the requirements map and method chunk process model. For every section in the requirements map we retrieve potentially required candidate chunks. After that, we analyse the process and product models of the retrieved chunks and select the more appropriate ones. Every selected method chunk must cover at least one section of the requirements map. Due to the lack of space, we do not detail here our method chunk selection and assembly process model. The details and examples of it can be found in [12]. We provide only the resulting method that we have obtained by using our assembly approach.

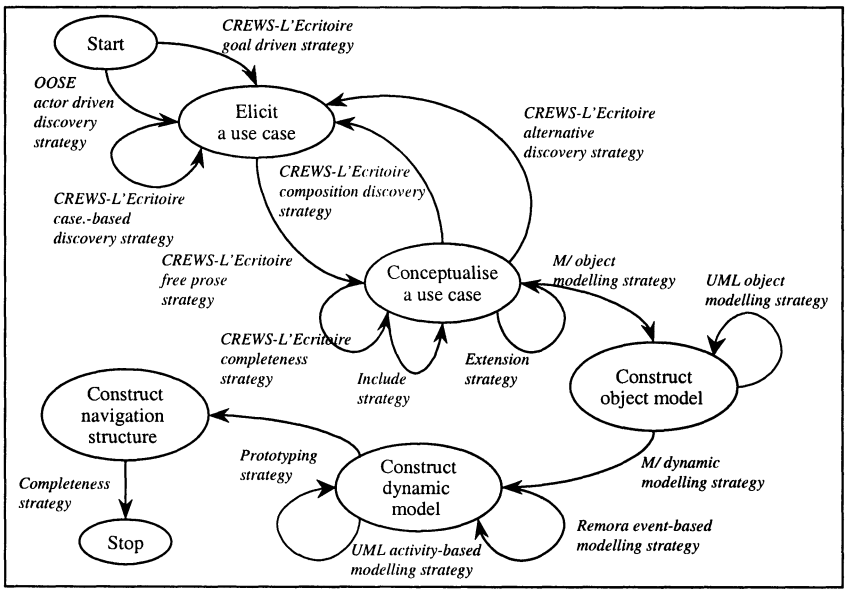

Figure 8. Example of the method matching the requirements map.

To cover requirements elicitation and conceptualisation process we have decided to use the *Use case model* proposed in [8]. In this model we have selected *Actor driven discovery strategy* for use case elicitation and *Include* and *Extension* strategies for use case conceptualisation. We have found that the guidelines for use case writing provided by this model were too poor to be integrated in our method. For this, we have selected the *CREWS-L'Ecritoire* [20, 21] method chunk providing more rich guidelines for scenario writing and conceptualisation. In this method we also selected chunks supporting elicitation of alternative and complementary use cases, variant and exception scenarios and validation of the obtained use case model. For the preliminary structural view construction we have selected object model from the *M7 method* [5] for its evolutionary characteristics whereas for the detailed object model we have selected UML [25] object diagram. In order to express the dynamic view of the system we have selected: *M7 dynamic model* (based on the Petri-net modelling approach) for objects life-cycle representation, *Remora* [17] for global system behaviour modelling and *UML activity diagram* for activity specification. Finally, the prototyping with Microsoft FrontPage was

used to construct the web site navigation structure. Figure 8 illustrates the obtained method process model.

4. INTENTION DRIVEN STRATEGY FOR REQUIREMENTS DEFINITION

In this paragraph we detail the guideline associated to the MEPM section <*Start, Specify method requirements, Intention driven strategy*> (Figure 1). This guideline helps the method engineer to specify which kind of adaptations the method must endure in order to satisfy the situation of the current project. A map, of a lower level of detail, expresses this guideline in Figure 9.

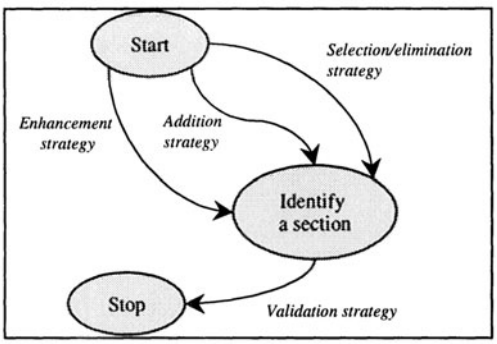

Figure 9. The intention driven requirements elicitation model.

We consider that the process model of every method can be expressed as a map with associated guidelines. (The method reengineering process model allowing to do that was presented in [13].) Therefore, the intention driven requirements elicitation process deals with the fundamental concepts of the map that is its sections. It is based on the analysis of the existing map sections and identification, which new sections must be added and which ones must be eliminated from the method map. That's why the key intention in the intention driven requirements elicitation map (Figure 9) is called *Identify a section.* The three strategies, *Enhancement, Addition and Selection/elimination*, supporting the achievement of this intention, correspond to the three methods adaptation cases presented in the paragraph 2 of this paper.

- The *Enhancement strategy* helps to identify the sections, which could be added into the method map with the objective to enrich the way of working proposed by the initial method by new manners to achieve some of the method intentions.
- The *Addition strategy* helps to identify the intentions, which could be added into the initial method map, and to specify the sections having these intentions as source and as target intentions.
- The *Selection/elimination strategy* helps to evaluate every section of the initial method map and to decide which sections must be kept in the adapted method and which ones could be eliminated from it.

Obviously, the three strategies may be combined to obtain the most suitable method for the situation at hand. The guidelines associated to these three strategies are introduced in the next sub-sections followed by an application exemple.

4.1 Enhancement Strategy

The *Enhancement strategy* deals this the method enrichment from the way of working perspective. It is relevant when the guidelines, proposed by the method, supporting the achievement of its intentions are inadequate or are not riche enough in the situation at hand. The guideline supporting this strategy is defined as follows:

```
G6: <Initial method map exists; Define a section to add
into the method map in order to enhance its way of
working>
    G6.1: <Initial method map exists; Identify a target
intention>
    G6.2: <Initial method map exists, a target intention
is identified; Discover strategies>
    G6.3: <Intention and strategies are defined; Define
sections>
```

This guideline proposes to identify first the intention, achievement of which must be enhanced by a new way of working (G6.1) and to discover potential strategies to attain this intention (G6.2). After that, the required sections can be defined and documented: for each identified strategy we need to discover first the precondition of its applicability that is the products required for the achievement of the target intention and to identify then the intention creating this product

(G6.3). This is a source intention in the section under construction. The application of this guideline is illustrated in the section 3.2.4.

4.2 Addition Strategy

The *Addition strategy* deals this the method enrichment from the functional perspective. It is appropriate when the method does not posses all the required models for the current IS project development. The guideline supporting this strategy is defined as follows:

```
G7: <Initial method map exists; Define sections to add
into the method map in order to complete its coverage>
    G7.1: <Initial method map exists; Define an intention
to introduce into the method map>
    G7.2: <A new intention is defined; Discover
strategies to achieve the new intention>
    G7.3: <A new intention is defined; Discover
strategies applying the result of the new intention>
    G7.4: <A new intention is defined, strategies are
defined; Define sections>
```

The process starts by the definition of the intention to include into the method map. Then, this intention must be attached to the rest of the method map that is to its other intentions by at least two strategies: one defining the manner to reach the new intention (G7.2) and one defining how the product, obtained after the execution of the new intention, could be applied for the achievement of the other intentions (G7.3). Therefore, at least two sections must be defined (G7.4): one containing the new intention as a target intention and the other containing this intention as a source intention. In the first case, we need to discover products necessary for the execution of the new intention following discovered strategies and to identify the intentions providing these products (the preconditions). In the second case, we should identify potential ways to apply the product constructed by the new intention and which intentions of the map will use this product as a source product (the post-conditions).

4.3 Selection/elimination strategy

The *Selection/elimination strategy* deals this the method restriction in order to keep in the method map only these sections, which are

relevant in the situation at hand. The guideline supporting this strategy is defined as follows:

```
G8: <Initial method map exists; Identify the sections to
keep in the method map>
    G8.1: <Initial method map exists; Select a section>
    G8.2: <A section is selected; Evaluate the relevance
of the section>
```

The objective of this guideline is to look at every section of the method map and to evaluate the relevance of its associated intention achievement guideline (IAG) in the current project situation.

4.4 Method adaptation exemple

As an application example, we propose to enhance the *Use case model* proposed in [8]. The map representing its process model is shown in Figure 10 (a). The details about the construction of this map can be found in [13].

It is evident that the use case conceptualisation task asks a lot of efforts and time. Moreover, the use case writing guidelines proposed by the authors of this model in [8] are not rich enough and are not really helpful. It is clear that not all scenarios necessitate to be written in the top level of details. It is possible that some use cases are less important or evident and a simple abstract could be sufficient to describe them whereas other use cases are essential and must be described very carefully. For that reason, we would like to reduce the effort required for the use case writing by classifying them and identifying the level of details necessary to their description. As a consequence, we decide to add a new intention into the use case model map that we call *Classify use cases*. To do that, we use the *Addition strategy* presented in the section 3.2.2. Following its guideline, we need to connect the defined intention with the other method intentions. The strategy required to reach this intention could be called *Organisation strategy*. The precondition for this strategy is the existence of at least one already elicited use case.

The classification of the use cases will influence their writing process. Therefore, the *Rank-based writing strategy* could connect the intentions *Classify use cases* and *Conceptualise use cases*. Moreover, the obtained use case classification could be helpful for the discovering other use cases, more or less important, of the lower or

higher level of abstraction. For this, we add another strategy that we call *Rank-based elicitation strategy* coming from the intention *Classify use cases* to the intention *Elicit use cases*.

Some use cases may perhaps ask several pages to be written completely. A graphical representation could be helpful to obtain a clear global view of a use case including all its alternatives and exceptions. For this reason, we propose to include a *Graphical representation strategy* representing another manner to conceptualise use cases. This may be done by applying the *Enhancement strategy* (section 3.2.1), which helps to add a new required section *<Conceptualise use cases, Conceptualise use cases, Graphical representation strategy>*.

Finally, we consider that the use case writing guidelines proposed by initial use case model authors could be eliminated from the final use case model map thanks to the *Selection/elimination strategy* (section 3.2.3). The obtained requirements map is shown in Figure 10 (b).

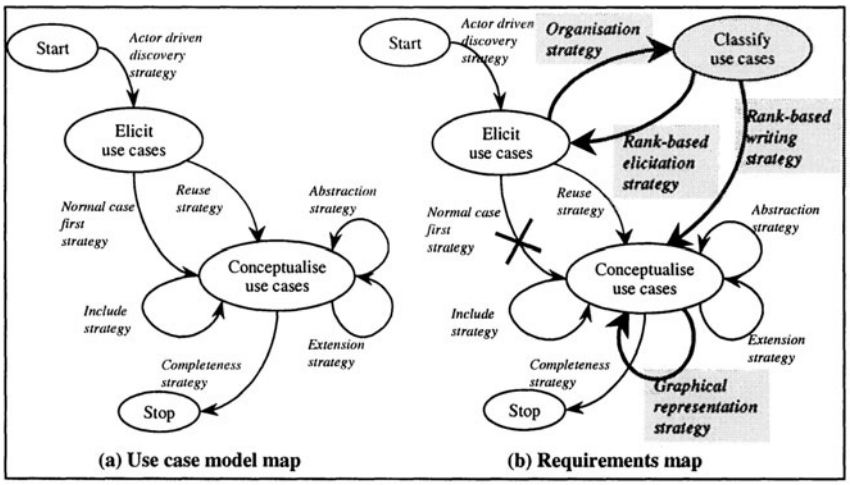

Figure 10. The use case model map.

The assembly process in this case consists in selecting at least one method chunk for each required section. In this example we select the use case classification and writing guidelines proposed by Cockburn [4]. This approach proposes two complementary use case

classification techniques: one is based on a three level goal hierarchy; other defines a design scope to capture in a use case typology. These two techniques cover the section <*Elicit use cases, Classify use cases, Organisation strategy*> in the requirements map. The guidelines supporting elicitation of other use cases of the lower or higher abstraction level are also provided by this approach and cover the section <*Classify use cases, Elicit use cases, Rank-based elicitation strategy*> in the requirements map. Moreover, this approach proposes different templates for use case writing as well as the content guidelines depending on the use case goal level and design scope. We decide to introduce this technique as a manner to write use cases. It covers the section <*Classify use cases, Conceptualise use cases, Rank-based writing strategy*>.

To cover the last section <*Conceptualise use cases, Conceptualise use cases, Graphical representation strategy*> of the requirements map, we select the approach proposed by Regnell [16], which helps to represent a complete use case as a graph called *usage view*. The resulting method map is illustrated in Figure 11.

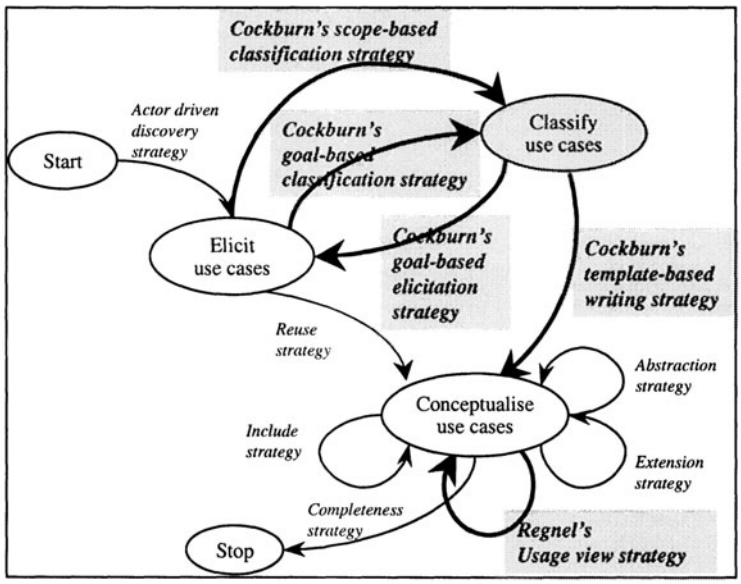

Figure 1. The adapted use case model map.

5. CONCLUSION

In this paper we look at situational method engineering from the method requirements elicitation perspective. To enable project-specific method construction we need to analyse the situation of the concerned project and to define the requirements for a method, which could be appropriate in this situation. For this reason, we propose an approach supporting the method requirements elicitation given specific project situation. The approach take into account the situation where a completely new method must be constructed as well as the possibility to adapt an existing method in different manners. The two requirements definition processes are represented as maps with associated guidelines. This allows us to offer flexibility to the method engineer for carrying out the requirements engineering activity. All the guidelines are detailed and illustrated in the paper.

Our approach has been applied in the IS development project with a Swiss watchmakers company. For this project we have developed a specific method by assembling chunks selected from different methods. The requirements for this method were defined by using our method requirements definition approach. The obtained results were satisfying and the experience was positive.

Our future preoccupation is the definition of the attributes to better specify the IS project situation. The identification of such characteristics could enrich our method requirements definition process as well as the method chunk selection process.

REFERENCES

1. Benjamen A., Une Approche Multi-démarches pour la modélisation des démarches méthodologiques. Thèse de doctorat en informatique de l'Université Paris 1, 1999.
2. Brinkkemper S., Method engineering: engineering of information systems development methods and tools. Information and Software Technology, Vol. 38, No.4, 1996.
3. Brinkkemper S., M. Saeki, F. Harmsen, Assembly Techniques for Method Engineering. 10th Conf. on Advanced Information Systems Engineering, CAiSE'98. Pisa Italy, 1998.
4. Cockburn A., Writing Effective Use Cases. Addison-Wesley, 2001.
5. Estier T., G. Falquet, J. Guyot, M. Léonard, Six Spaces for global Information Systems Design. Proc. of IFIP Working Conference on the Object-Oriented Approach in Information Systems, Québec, Canada, October 1991.

6. Harmsen A.F., S. Brinkkemper, H. Oei, Situational Method Engineering for Information System Projects. In Olle T.W. and A.A. Verrijn Stuart (Eds.), Mathods and Associated Tools for the Information Systems Life Cycle, Proc. of the IFIP WG8.1 Working Conference CRIS'94, pp. 169-194, North-Holland, Amsterdam, 1994.
7. Harmsen A.F., Situational Method Engineering. Moret Ernst & Young , 1997.
8. Jacobson I., M. Christerson, P. Jonsson, G. Oevergaard, Object Oriented Software Engineering: a Use Case Driven Approach, Addison-Wesley, 1992.
9. Plihon V., 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. Proc. of the Int. Software Process Association's 5th Int. Conf. on Software Process (ICSP'98), Chicago, Illinois, US, 1998.
10. Prat N., Goal Formalisation and Classification for Requirements Engineering. Proceedings of the Third International Workshop on Requirements Engineering: Foundations of Software Quality REFSQ'97 , Barcelona, June 1997.
11. Punter H.T., K. Lemmen, The MEMA model : Towards a new approach for Method Engineering. Information and Software Technology, 38(4), pp.295-305, 1996.
12. Ralyté J., C. Rolland, An Assembly Process Model for Method Engineering. Proceedings of the 13th Conference on Advanced Information Systems Engineering, CAISE'01, Interlaken, Switzerland, June 6 – 8 2001.
13. Ralyté J., C. Rolland, An approach for method reengineering. Proceedings of the 20th International Conference on Conceptual Modeling, ER2001, Yokohama, Japan, November 27-30 2001.
14. Ralyté J., C. Rolland, V. Plihon, Method Enhancement by Scenario Based Techniques. 11th Conf. on Advanced Information Systems Engineering CAiSE'99, Germany, 1999.
15. Ralyté J., Reusing Scenario Based Approaches in Requirement Engineering Methods: CREWS Method Base. Proceedings of the 10th International Workshop on Database and Expert Systems Applications (DEXA'99), 1st International Workshop on the Requirements Engineering Process – Innovative Techniques, Models, Tools to support the RE Process (REP'99). Florence, Italy, 1999.
16. Regnell B., K. Kimbler, A. Wesslen, Improving the Use Case Driven Approach to Requirements Engineering. I. C. S. Press, Eds., Second IEEE International Symposium On Requirements Engineering, (York, England), 1995.
17. Rolland C., O. Foucaut, G. Benci, Conception des Systèmes d'Information, la méthode Remora. Eyrolles, 1987.
18. Rolland C., N. Prakash, A proposal for context-specific method engineering. IFIP WG 8.1 Conf. on Method Engineering, Chapman and Hall, pp 191-208, Atlanta, Gerorgie, USA, 1996.
19. Rolland C., V. Plihon, J. Ralyté, Specifying the reuse context of Scenario Method Chunks. Proceedings of the 10th International Conference on Advanced Information System Engineering (CAISE'98), Pisa, Italy, 1998.
20. Rolland C., C. Souveyet, C. Ben Achour, Guiding Goal Modelling Using Scenarios. IEEE Transactions on Software Engineering, special issue on Scenario Management, Vol 24, No 12, 1998.

21. Rolland C., C. Ben Achour, Guiding the construction of textual use case specifications. Data & Knowledge Engineering Journal, 25(1), pp. 125-160, March 1998.
22. Rolland C., N. Prakash, A. Benjamen, A multi-model view of process modelling. Requirements Engineering Journal, p. 169-187,1999.
23. Saeki M., K. Iguchi, K Wen-yin, M Shinohara, A meta-model for representing software specification & design methods. Proc. of the IFIP WG8.1 Conference on Information Systems Development Process, Come, pp 149-166, 1993.
24. Song X., A Framework for Understanding the Integration of Design Methodologies. ACM SIGSOFT Software Engineering Notes, 20 (1), pp. 46-54, 1995.
25. Rational Software Corporation, Unified Modelling Language version 1.3. Available at http://www.rational.com/uml/resources/documentation/, 2000.
26. van Slooten K., S. Brinkkemper, A Method Engineering Approach to Information Systems Development. In Information Systems Development process, N. Prakash, C. Rolland, B. Pernici (Eds.), Elsevier Science Publishers B.V. (North-Holand), 1993.