MADISE: Method Engineering-based Approach for Enhancing Decision-Making in Information Systems Engineering

Elena Kornyshova

MADISE: Method Engineering-based Approach for Enhancing Decision-Making in Information Systems Engineering

Elena KORNYSHOVA

A thesis submitted to the faculty of the University of Paris 1 - Panthéon Sorbonne

in partial fulfillment of the requirements for the degree of Doctor of Philosophy (Ph.D.)

in Computer Science

2011

Paris, France

Jury

Prof. Colette Rolland

Prof. Camille Salinesi

Dr. Rébecca Denekère

Prof. Oscar Pastor

Prof. Naveen Prakash

Prof. Regine Laleau

To my Family

Acknowledgements

I would like to thank all people who have helped and inspired me during my doctoral study.

First and foremost, I want to thank Colette Rolland (Université de Paris 1 Panthéon-Sorbonne, France) for being the advisor of my research through this long path, for her guidance and support during my work. Her wide knowledge and her logical way of thinking have been of great value for me. Her understanding, encouraging and personal guidance have provided a good basis for the present thesis.

I am also very grateful to Camille Salinesi (Université de Paris 1 Panthéon-Sorbonne, France) and Rébecca Deneckère (Université de Paris 1 Panthéon-Sorbonne, France), their suggestions and feedbacks have been very valuable in the development of my research. I want to thank them for being always ready for discussions and for their suggestive remarks.

I address my greatest thanks to the other members of my PhD committee. First of all, my dearest thanks to my PhD reviewers Pr. Oscar Pastor (Centro de Investigación en Métodos de Producción de Software PROS, Valence, Espagne) and Pr. Naveen Prakash (Delhi Institute of Technology, Delhi, Inde) who dedicated their time to read about my research and to participate in my inquiry. Thanks also to Pr. Régine Laleau (Université Paris-Est Créteil, France) who kindly accepted to be a part of this inquiry.

During these years, I have been sharing CRI with many people: Carine, Charlotte, Manuele, Daniel, Selmin, Irina, Said, Judith, Bruno, Kahina, Adrian, Oumaima, Sana, Salma, Raul, Elena I, Laure-Hélène, Assia, Kadan, Olfa, Anne, Ines, Rim, Hicham, Ramzi, Islem, and Yves-Roger. Some of them have become friends, especially Bruno, Kahina, Adrian, and Rébecca. Thanks you all for the time spent together, the long scientific (and non-scientific) discussions, and all the fun we had through these last years.

Finally, I want to give very special thanks to my family for they support, encouragement and, above all, their love. My parents, of course, who saw me going through two successive thesis! They have lost a lot due to my research abroad. Without their encouragement and understanding it would have been impossible for me to finish this work. And last but not the least, Emeric, who was next to me with his support, encouragements, help, and care for me.

Abstract

The goal of this work is to offer an approach for enhancing decision-making in information systems engineering based on the Method Engineering principles. In order to achieve this goal, we have developed the *MADISE* (***MA***ke ***D***ecisions in ***I***nformation ***S***ystems ***E***ngineering) approach.

The MADISE approach includes four following elements: the DM ontology, the DM method family, the MADISE configuration process, and the MADISE methodological repository.

The DM Ontology (DMO) is an ontology of decision-making knowledge, which allows representing DM concepts in a structured manner. The main goal of DMO is formalizing DM situation and specifying requirements for DM.

The DM Method Family is a collection of DM method components organized into a family for their easier usage in practice. DM components represent detailed guidelines for DM activities associated to the specific context of their use.

The Method Family Configuration Process is a ME process helping IS engineers to customize DM method family. This process guides the selection and assembly of DM components into an application method adapted to a given situation.

The Methodological Repository is a repository, which provides IS engineers with a methodological support for realizing DM activities.

For the needs of the MADISE approach, some fundamentals of the ME science are developed. Firstly, the concept of method family is proposed. Secondly, an approach for contextualization of method components is developed. Thirdly, an indicator-based guidance methodology is suggested for customizing method families according to a given situation. These elements are applied to DM methods in order to achieve the main research goal.

Résumé

L'objectif de cette thèse est de proposer une approche (*MADISE* – ***MA***ke ***D***ecisions in ***I***nformation ***S***ystems ***E***ngineering), pour améliorer la prise de décision (DM – Decision-making) dans l'ingénierie des systèmes d'information, basée sur les principes d'ingénierie de méthodes.

Cette approche comprend quatre éléments centrés sur la prise de décision : une ontologie, une famille de méthodes, un processus de configuration et un référentiel méthodologique.

L'ontologie de prise de décisions (DMO) est une ontologie de connaissances sur la prise de décision. Son objectif principal est de formaliser la situation de DM et les exigences pour la DM.

La famille de méthodes de méthodes de prise de décision est une collection de composants de méthodes de prise de décisions. Les composants décrivent les activités de prise de décision associées au contexte spécifique de leur utilisation.

Le processus de configuration de famille de méthodes est un processus qui accompagne les ingénieurs des SI dans l'application de la famille de méthodes. Ce processus guide la sélection et le montage des composants dans une méthode d'application adaptée à une situation donnée.

Le référentiel méthodologique fournit aux ingénieurs des SI un support méthodologique pour la réalisation des activités de DM.

Pour les besoins de l'approche MADISE, certains principes de l'ingénierie des méthodes ont été développés : la notion de famille de méthodes, une approche pour la contextualisation des composants de méthode et une approche basée sur des indicateurs pour la configuration des familles en fonction des situations. Ces éléments sont appliqués à des méthodes de prise de décisions afin d'atteindre les objectifs principaux de recherche.

Contents

Part I. Research Problem

Information system (IS) development processes were traditionally organized into sequences of activities with a rigid control flow. However, in practice, IS engineers are faced with countless methodical choices: they can choose to execute one activity completely or partially, to combine aspects of two or more different activities, or to ignore an activity. That means that any IS development process may be seen as a sequence of decisions made to enact a unique process adapted to a specific situation. Chapter 1 explores this statement in order to define the key issues to consider for establishing the research questions this thesis intends to answer.

There are many methods of decision-making (DM). They come from the operational research field and deal with the decision making process in different ways.

Chapter 2 defines fundamentals about DM and offers a review of the main DM methods. This chapter also addresses more particularly the question of DM in IS Engineering (ISE) in order to show that several problems arise in the application of DM methods in this field. This conclusion leads to the establishment of requirements that a DM approach in ISE should satisfy.

These requirements are taken into account in a new proposed approach called MADISE. Chapter 3 gives an overview of this approach and offers a synthesis of the contributions of this thesis.

Chapter 1: Introduction

1.1. Thesis Context

Nowadays, Information Systems (IS) are a central element for helping organizations functioning and providing them with a work environment increasing their competitive advantages. In these conditions, IS must be elaborated in such a way that they (i) are optimized with regard to the used resources, (ii) take into account the specific situation and (iii) bring together different and often conflicting stakeholders' interests.

Information System Engineering (ISE) is a science which describes processes of IS analysis, design, development, implementation, maintenance, and evolution. It contains different activities that offer guidelines to IS engineers for elaborating IS. In traditional ISE processes, activities are grouped into phases (analysis, design, coding, testing) [Boehm, 1988] [Royce, 1970], which are organized into predefined sequences.

Faced to these processes in practice, IS engineers deal with different situations, in which a choice must be made. This choice may concern, for instance, a goal, an action, or a tool. [Ralph *et al.*, 2008] highlights the fact that, in practice, the software developer is faced with a myriad of methodical choices as he can choose to execute one activity completely or partially, combine aspects of two or more different activities, or ignore another one. The decisions made through the process execution lead the IS engineer to enact a process which is unique and suited to the situation at hand. In fact, information system design, development, implementation, and every other processes in ISE can be seen as a set of steps of two kinds: first, of decision-making (DM) (where a decision is made) and, second, of decision performing (when the decision result is applied for a given purpose). In this manner, all ISE processes during their execution include steps where several alternatives call for decisions, from operational through tactical to strategic ones. Researches on several engineering fields (systems engineering [Ruhe, 2003], process engineering [Rolland *et al.*, 2000], method engineering

[Aydin, 2006], and so on) show that there are many development cases where IS engineers has critical choices to carry out. Several examples of decisions covering all phases of the software life cycle can be cited:

- To invest or not to invest in new software;
- To select between an open or a proprietary tool;
- To prioritize requirements during the analysis phase;
- To select an appropriate architecture or a COTS during the design phase;
- To decide whether a new component should be integrated during the maintenance phase;
- To arbitrate between discarding and maintaining a software application.

Though, IS engineers meet decisions from very simple to very complex ones at different phases of ISE. IS engineers must consider this kind of problems as a decision-making problem.

A decision is an act of intellectual effort initiated for satisfying a purpose and allowing a judgment about the potential actions' set in order to prescribe a final action. A DM problem is firstly defined by the presence of alternatives (potential actions). The concept of alternative designates the decision object. Any decision involves at least two alternatives that must be well identified. The prescription is based on the study of several aspects characterizing alternatives, which are criteria. A simple approach consists in using one criterion to carry out the selection between alternatives. The traditional example is the selection of projects according to the Net Present Value (NPV). However, using a single criterion is not sufficient when the consequences of the alternatives to be analyzed are important [Roy *et al.*, 1993] [Roy, 2005], when there are multiple viewpoints and contradictory criteria. Multicriteria analysis, in contrast to a monocriterion approach, allows a more in-depth analysis of a problem because it aims at considering various aspects. The goal of multicriteria DM (MCDM) methods is to define priorities among alternatives (actions, scenarios, projects) according to multiple criteria. These methods deal with indicators having different nature (quantitative or qualitative). However, they are more complicated as they must be aggregated into a general value or function [Roy, 1996] [Keeney *et al.*, 1993].

MCDM methods have shown their qualities for over 40 years [Berander, 2005] and they currently dominate in the field of decision-making [Baudry *et al.*, 2002] [Gomez-Limon *et al.*, 2003]. They appeared at the beginning of the Sixties, and their number and application contexts increase continually. We can mention five main families of MCDM methods: MAUT (Multiattribute Utility Theory) [Keeney *et al.*, 1993], AHP (Analytic Hierarchy Process) [Saaty, 1980], outranking methods [Roy, 1996], weighting methods [Keeney, 1999], and fuzzy methods [Fuller *et al.*, 1996].

With regard to ISE, the issue of DM has already been explored with respect to requirements engineering [Ngo-The *et al.*, 2005], to method engineering [Aydin, 2006], and, more generally, to systems engineering [Ruhe, 2003]. For instance, the GRL (Goal-oriented Requirement Language) model allows evaluating solutions according to their contribution to the goals [Amyot *et al.*, 2002]. Ruhe emphasizes the importance of DM in SE along the whole life cycle [Ruhe, 2003]. Several examples of different DM methods application can also be mentioned: AHP for prioritizing

requirements [Maiden *et al.*, 2002] and evolution scenarios [Papadacci *et al.*, 2005]; Saeki uses weighting method to deal with software metrics [Saeki, 2003], and so on.

1.2. Problem Statement

Regarding to DM in ISE, the following findings can be made.

- **Many decisions are made in the ISE field.** ISE processes are teleological by their nature [Loucopoulos *et al.*, 1995]. That means these processes contain steps where decisions must be made. Traditional ISE processes do not take into account a full decision-making stage. They give hints about how to make a decision but there is usually no formalization of the complete DM process. Sometimes, the ISE methods offer guidance to help engineers through these choices. However, in many cases, decisions are carried out in a poor way: the choice is often intuitive and thereafter hazardous with unpredictable consequences [NgoThe *et al.*, 2005] [Ruhe, 2003]. As shown in [Ruhe, 2003], engineering-related decisions may result from the need to satisfy practical constraints such as quality, cost or time. Ruhe stresses the importance of DM in the field of IS because of: (i) time, effort, quality and resources constraints; (ii) presence of multiple objectives; (iii) uncertain, incomplete and fuzzy information, and (iv) complex decision space. Considering these aspects implies that the number of researches dealing with DM in ISE increasingly grows. For instance, the multicriteria DM methods spread throughout Computer Science [Kou *et al.*, 2009].
- **Many methods are available to support DM in a systematic way.** The operational research domain has produced different methods – from very simple ones to very complex ones – that could be adequately used in these situations. Using these methods should not only facilitate DM activities, but also it would allow considering specific situation, better involving stakeholders, increasing their confidence in the final decisions and so on. However, these methods have different characteristics such as complexity or ability to deal with quantitative or qualitative criteria. Moreover, these methods are more or less efficient in function of the situation in which they are applied. Unfortunately, there is no universal method able to deal with all DM problems. Each DM method is suitable for problems with specific characteristics such as, for instance, the number and nature of alternatives and of decision criteria, the presence of multiple stakeholders with different viewpoints.
- **DM methods application in ISE is still limited.** Several reasons of such limitation could be mentioned. Firstly, DM methods coming from the operational research are not expressed in terms used in the IS field. Secondly, IS engineers are confronted with some practical difficulties. For instance, no work has been done to understand how, when, or which of these methods could be used during ISE activities. Often DM methods are selected arbitrarily. Sometimes the analyst uses a method because he/she is already familiar with it. Other times a method is developed in an “ad hoc” way. It also happens that a method is chosen simply because a software which supports it is available at the right time [Laaribi, 2000] [Ulengin *et al.*, 2000]. In addition, the existing works in the ISE research aim at resolving a DM problem each time.

The lack of DM in ISE can be considered at three levels: (i) at the model level, (ii) at the method level, (iii) and at the tool level. At the model level, decisions are often ill formulated. At the method level, intuitive and ad hoc decisions overshadow the method-based ones. At the tool level, even if DM tools exist, there is none with a complete context-aware DM process. This thesis addresses the lack of DM in ISE at the model and method levels.

At the ***model level***, many decisions are made in the field of ISE. Nevertheless, decisions are not formalized in terms of alternatives and criteria; their consequences are not analyzed. Decisions are not transparent and are characterized by poor understanding and describing decision problems, by the lack of transparency and of empirically evaluated models. Stakeholders' contradictory interests are not always considered.

With regard to the ***method level***, the situation is common in ISE when decisions are made in an intuitive manner or based on the engineer experience and expertise. Even if they are present, the scientifically founded decision-making methods are not very widespread in this field. There are needs for taking into account the situation at hand, selecting an adequate decision-making method and, if necessary, adapting it and integrating it into existing methodologies.

Three common situations of DM in ISE can illustrate these concerns.

- **Several DM methods exist for resolving the same DM problem.** This is a case, for instance, of the requirements engineering field. There are many methods dealing with requirements prioritization, for instance, Cost-Value Approach [Karlsson *et al.*, 1997], Prioritization Matrix [Wiegers, 1999], Requirement Prioritization Tool [Moisiadis, 2002], WinWin [Ruhe *et al.*, 2003], and REDEPEND [Maiden *et al.*, 2002]. Requirements prioritization methods are from very simple (like bubble soft) to very complex ones (for example, the WinWin approach [Ruhe *et al.*, 2003]). Some attempts are made to compare different methods in this field. However, an approach allowing selecting one of them in a given situation does not exist. In this case, an IS engineer is faced with a problem of how to select a DM method.
- **Some DM methods exist, but the associated guidelines are not sufficient to resolve the DM problem.** Therefore, in this kind of guidelines, there is very little information about how to achieve decisions. This is the case, for instance, of the Rational Unified Process (RUP) [Kruchten, 1998] [RUP, 2007]. RUP is a body of software engineering practices, which is maintained on a regular basis to reflect changes in industry practices. Many RUP practices call for decision-making, as select and acquire tools, prioritize use cases, analyze and prioritize risks, and so on. For instance, the *Select and Acquire Tools* task guides the adoption of tools that support other tasks in the RUP. Tools that need to be selected should fit the particular requirements of the organization for which the selection is made. Furthermore, special tools sometimes have to be developed internally to support special needs. One of the steps in this task is to collect information about tools. This information serves as selection criteria to help the system engineer to decide which tool is right for the project at hand. The criteria for tool selection are tool features, vendor and cost characteristics. The RUP proposes to grade each criterion for evaluating candidate tools. However, the guidance stops there and the IS engineer is left alone at the moment of the actual decision-making.

- **No DM method exists to deal with a given problem.** There are many cases in ISE, when a suitable method is not suggested because DM problems are not well formulated or even are not viewed as having the DM nature. For instance, some new researches deal with variability in process models [Derguech *et al.*, 2010] [Santos *et al.*, 2010]. In [Derguech *et al.*, 2010], an indexing structure is proposed for the process models configuration taking into account different process variants. [Santos *et al.*, 2010] consider variations in business processes modeled with Business Process Modeling Notation (BPMN). Variations represent alternative paths of execution in a workflow. In order to configure a process and to select a path, the authors suggest relating process variants to non-functional requirements (NFR). However, the variations are not analyzed as a DM problem, and only a hint about how to make the selection is given. In two cases, detailed guidelines for DM are needed.

In that way, many ISE methodologies must be improved. The main DM problems in ISE can be resumed as follows: DM situations and requirements for DM are ill-formulated; DM methods are described in different manners which do not allow deciding whether their application in a given situation is suitable. These DM problems could be resolved if DM is improved at both model and method levels.

Moreover, several issues must be considered:

- **Variability Issue.** [Taylor, 1964] states that thevariability acquisition allows discovering variation points in a problem. Variation points' existence requires that some solutions must be available to deal with them. The suggested approach has to take into account variability in ISE and to provide a solution for selecting a variant from the available ones.
- **Context-awareness Issue.** Context-awareness means that ISE processes are executed with consideration of the situation at hand [Salber *et al.*, 1998]. Decision-making support must be adapted to the specific context.
- **Conflict Resolution Issue.** Many people – stakeholders and other participants – are involved in the ISE processes. Their interests and points of view are often contradictory. The "dispute resolution" must be taken into account in ISE processes [Osterweil *et al.*, 2010]. For the same reason, DM processes must also contribute to the conflict resolution.
- **Goal Orientation Issue.** Since Veblen [Veblen, 1898], the teleological principles integrate the agent's intentions in processes. In contrast to traditional development processes, modern ISE processes must be oriented on achieving the intentions (goals, objectives) associated to the result that the engineer wants to achieve [Loucopoulos *et al.*, 1995]. In this way, the suggested approach must consider intentions, which participants of the DM process have for resolving a given problem.
- **Reusability Issue.** The reuse principle consists in constructing methods based on existing methods or method components conceived earlier. It allows decreasing costs and time in ISE. DM activities must also be based on this principle, as the application of DM methods has to be optimized.

1.3. Research Questions

Based on the previous sections, the main research question in this thesis is stated as follows:

> ***Main RQ:* How methodological knowledge about decision-making can be represented to facilitate the inclusion of decision making processes into ISE methods?**

This research aims at providing methodological support to help IS engineers in their DM activities. For responding to this main research question, we must answer to the following research questions:

> *RQ1: How to identify a decision-making situation and to formalize requirements for decision-making?*

The answer to this research question must allow defining the DM concepts, which are necessary to formalize all elements of the situation requiring DM activities. Coming from the decision-making science, they should be appropriated to the IS engineering field.

> *RQ2: How decision-making methods should be described?*

The purpose of this question is to find a way of describing DM methods which enables their compatibility with IS engineering methodologies and their context-aware retrieval.

> *RQ3: How to construct a context-aware decision-making method in a given situation?*

The goal is to define a process, which allows selecting and/or constructing a decision-making method by formalizing DM requirements and considering the concrete situation.

> *RQ4: Which kind of support can be offered to IS engineers for improving their methodologies by decision-making?*

This question aims at creating a kind of support for the "user friendly" application of the suggested approach.

In order to deal with the above-mentioned issues (Section 1.2), the suggested approach has to fulfill the following requirements:

- To be ***generic*** as it can be applied to any process of IS engineering, including both enhancing existing IS engineering processes by DM functionalities and creating new DM methods;
- To be ***context-aware*** as it must be adapted to each specific situation of IS engineering;
- To be ***human-aware***. That means to guide IS engineers in DM activities and to take into account the contradictory interests of stakeholders and other participants;
- To be ***goal-oriented*** as it must satisfy intentions that IS engineers have in DM processes;
- To be based on the principle of ***reusability*** as it allows the usage of the same DM methods in different situations.

1.4. Thesis Contributions

Our proposal consists in an approach that helps decision makers to consider the different aspects of the problem at hand and their goals for constructing a DM method adapted to a given situation. This method can be applied as a whole or be integrated into an existing ISE methodology.

In order to achieve this goal, we have developed the *MADISE* (***MA***ke ***D***ecisions in ***I***nformation ***S***ystems ***E***ngineering) approach. Our proposal uses the Method Engineering fundamentals in order to make compatible DM methods with ISE methodologies. Method Engineering (ME) is the discipline to design, construct and adapt methods, techniques and tools for the IS engineering [Brinkkemper, 1996] [Rolland, 2005].

ME approaches, and in particular, situational ME approaches, highlight some very interesting aspects to improve the use of DM methods: modularity, reuse, flexibility, and situation-awareness. First of all, they consider the methods as *modular*, which means that a method may be divided into blocks (method components) that can be recomposed together following the needs of the project at hand. This blocks composition helps to improve the *reuse* of the method, as each block may be used several times. The reuse principle allows constructing methods based on existing methods or method blocks conceived earlier. This combination of modularity and reuse aims to give more *flexibility* to the method engineer to construct methods adapted to a specific project. According to [Harmsen *et al.*, 1994], the flexibility can vary from rigid methods to modular methods. This is the reason why ME approaches are considered as adapted to concrete situations and are called *situation-aware.* We apply the fundamentals of the ME science to DM methods in order to enhance their usability, context-awareness, reuse, and interoperability with IS.

Thereby, the goal of this work is to offer an approach for enhancing decision-making in information systems engineering based on the Method Engineering principles.

For the needs of the MADISE approach, some fundamentals of the ME science are developed in this thesis. Firstly, the concept of method family is proposed. Secondly, an approach for contextualization of method components is developed. Thirdly, an indicator-based guidance methodology is suggested for customizing method families according to a given situation. These elements are applied to DM methods in order to achieve the main research goal.

The MADISE approach includes four following elements: the DM ontology, the DM method family, the MADISE process, and the MADISE methodological repository. These elements aim at responding to the research question defined in Section 1.3.

- **The DM Ontology** (DMO) is an ontology of decision-making knowledge, which allows representing DM concepts in a structured manner. The main goal of DMO is formalizing DM situations and specifying requirements for DM (Research Question RQ1).
- **The DM Method Family** is a collection of DM method components organized into a family for their easier usage in practice. DM components represent detailed guidelines for DM activities associated to the specific context of their use (Research Question RQ2).

- **The Method Family Configuration Process** is a ME process helping IS engineers to customize DM method family. This process guides the selection and assembly of DM components into an application method adapted to a given situation (Research Question RQ3).
- **The Methodological Repository** is a repository, which provides IS engineers with a methodological support for realizing DM activities (Research Question RQ4).

1.5. Thesis Outline

This work is organized into four parts:

Part I – Research Problem: The goal of this part is to introduce the problem of decision-making in ISE. It contains three chapters. After an introduction to this thesis (Chapter 1), this Part contains a State-of-the-Art on decision-making and its application in information system engineering (Chapter 2). It is closed by an overview of the MADISE approach (Chapter 3).

Part II – Decision-Making Ontology: This part aims at defining the DM ontology, which models and formalizes DM knowledge. The DMO purpose is twofold: (i) formalization of DM methods and method components in a standardized manner and (ii) formalization of requirements for DM in a given situation for the further selection of the corresponding DM components and the construction of a situational DM method. DMO constitutes a result used in other steps of the MADISE approach such as the DM method family constitution and configuration. Therefore, we have described DMO in an independent part including one chapter.

Part III – Decision-Making Method Family: The goal of this part is to describe the family of the DM methods and the process of its configuration. It includes four chapters. Chapter 1 represents a State-of-the-Art on Situational Method Engineering approaches that we have analyzed in order to select the most suitable to DM methods approach. Chapter 2 offers a detailed representation of the DM method family. In Chapter 3, we detail the process of contextualization of method components and method families and apply it to the MADISE family. Finally, Chapter 4 emphasizes the process of method family configuration to construct application methods adapted to the concrete situations and its application to our case.

Part IV – MADISE Approach in Practice: The last part's goal is to show how the MADISE approach could be used in practice. It includes two chapters. Chapter 1 describes the tool designed for supporting the MADISE approach and two cases of the MADISE approach validation. We conclude our thesis by exploring the obtained results and future research in Chapter 2.

Chapter 2: State-of-the-Art on Decision-Making and DM in IS Engineering

2.1. Introduction

Decision-making is very important for an organization as a whole, and for their IS in particular. The key of the competitiveness consists in the ability to detect problems and to resolve them. Each decision aims at resolving a problem. A decision is good if it allows the maximal goals achievement and minimal resources use. Appropriate and taken at the right time decisions enable a complementary competitive advantage. However, the actual situation is characterized by the uncertainty, rapidly evolving environment, and competitors' innovations. This implies the requirement for decision-making activities that must be flexible and dynamically constructed. These issues are generic to the management field, but they are also available for IS engineering. Enterprise IS is a framework for other functions of management and the correct functioning of IS is a pledge of success for enterprise.

The operational research domain has produced many methods – from very simple ones to very complex ones – that could be adequately used in these situations. Decision-making may be considered as an outcome of a cognitive process leading to the selection of an action among several alternatives. It might be regarded as a problem solving activity, which is terminated when a satisfactory solution is found.

This role of decision-making implies the necessity to study firstly the nature of the decision-making problem. We analyze then basic information about decision-making. The next step is to motivate the usefulness of DM in ISE. Lastly, we offer a detailed State-of-the-Art of the approaches allowing selecting DM methods.

More precisely, this Chapter offers a State-of-the-Art on decision-making and includes the following elements:

- Explanation of the decision-making as a way of resolving problems;
- Fundamental notions of decision-making;
- DM in Information System Engineering;
- State-of-the-Art on the selection of DM methods.

We conclude this Chapter by analyzing problems of DM in ISE.

2.2. Decision-Making as a Mean for Resolving Problems

Decision-making is based on information about the ***problem situation***. The notion of problem or problem situation is used for describing the gap between the desired situation (goals) and the actual situation. Problem resolution means to reduce this gap and consists of selecting one of several possible ways to move from the current situation to the target situation. The situation must also be considered in the DM process. Problem situation and other factors (often external) affect the decision-making process. Knowledge about different factors, impacting both the problem situation and the DM process, allows taking more precise decisions. Thus, available information determines decisions quality. For better handling information, it must be presented in a structured manner, which is accessible for different participants.

Problem situations in a discrete case[1] can be classified into three types according to three possible way of solving: one mandatory solution, one optional solution, and at least two solutions (cf. Figure I.2.1.).

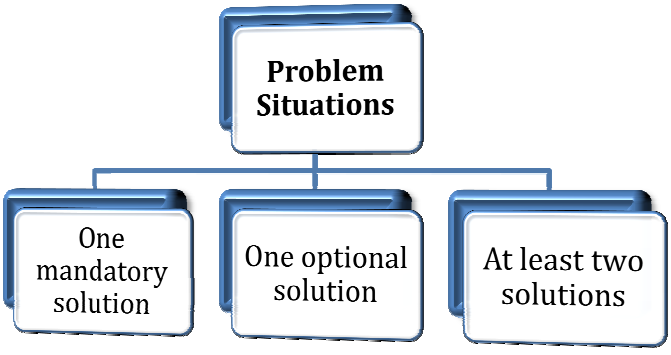


Figure I.2.1. Classification of Problem Situations according to the Solution Type.

In the first case, the problem resolution consists in exploring a possible way to solve the given problem and in applying this solution. Only one possible way to solve the problem is found and this

[1] Discrete DM deals with the finite alternatives number instead of continuous DM, which implies infinite number of alternatives. In ISE, finite number concerns most DM situations. For this reason, we concentrate our State-of-the-Art on discrete DM.

solution is mandatory for application. Thus, a decision is not to be made. On the contrary, one optional solution implies that a decision must be made. This decision is formulated like: to do or not to do. In the third case, several solutions are available and the decision-maker must arbitrate between them.

Decision-making appears only if a choice must be made. Therefore, decisions are made in the second and third situations. We qualify these types of situations as ***decision-making situations*** (See Figure I.2.2.).

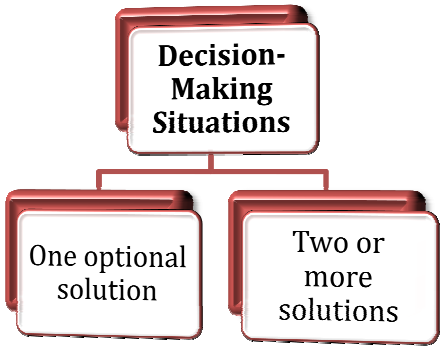


Figure I.2.2. Decision-Making Situations.

As the problem is the discrepancy between the actual situation and the target situation, the decision aims at decreasing this gap. Resolution of a problem situation can be a typical task, which is repetitive in a given area, or a unique task, which is new.

Another important term is the DM process. If a decision is a product, decision-making is a process. A ***decision-making process*** starts when a problem situation arises and stops when actions for the problem removal have been taken. The decision-making process includes three mandatory elements: elicitation of the problem situation, a set of possible solutions, and the elaboration of a decision using a ***DM method*** with the participation of an individual (or a group) ***decision-maker***. Thus, the decision-making process includes not only the choice but also others steps, like identification of possible alternatives.

The decision-maker is responsible for decisions and his mental activity is characterized by the following: possibility to choose between several alternatives, goals' presence, and volitional act as he generates a decision through different motivations and opinions. There are situations requiring expertise, experience, and even creativity of the decision maker. This implies decisions made in an intuitive manner. The nature of these decisions is flexible and adaptive. They are taken with the shortest delay. Another kind of decisions is method-based. These decisions are based on scientific methods and often on mathematical models.

[Afonotchkin *et al.*, 2009] mentions different factors of decision quality:

- Complexity of the DM situation,

- Available information about the problem,
- Skills and experience of decision makers,
- Resources and means engaged for decision-making,
- DM method,
- Organizational structure of the DM unit (decision-makers number and hierarchical relationships).

In order to enhance the decisions' quality, each decision maker could be assisted. [Roy 1985] defines decision-aid as follows:

> *"Decision-aid is the activity of one who uses explicit – but not necessarily completely formalized – models to obtain elements of answers to questions raised by an actor involved in a decision process. These elements tend to clarify the decision and, usually, to prescribe or simply to encourage behavior that will increase the coherence between the evolution of the process and the objectives supported by this actor".*

Thereby, a decision is an act of intellectual effort initiated for satisfying a purpose and allowing a judgement about the potential actions set in order to prescribe a final action. The main elements of decisions and decision-making are presented in the following section.

2.3. Decision-Making Fundamentals

This section offers an overview of the fundamentals of the DM science, namely main DM notions, DM actors, process, and main DM methods. We start by explaining several DM notions using an example.

2.3.1. Presentation with an Example

Before beginning the detailed presentation of the main DM notions, we introduce some ideas based on an illustrative example. For this purpose, we use a case that deals with information system (IS) security within the requirements engineering (RE) field.

Assume that an engineer must analyse IS security. He could use several RE methods. The project at hand is characterised by:

- a great impact on the whole organisation;
- the need for ensuring the greater progress;
- the absence of experts of this field in the organisation that does not plan to employ any of them;
- the need for a better explanation of methods and their application.

The panel of the available RE methods includes:

- Method 1 (M_1) – NFR Framework [Chung et al., 1999],
- Method 2 (M_2) – KAOS [Dardenne *et al.*, 1993],
- Method 3 (M_3) – Secure Tropos [Bresciani *et al.*, 2004],
- Method 4 (M_4) – GBRAM [Anton, 1997], and
- Method 5 (M_5) – Misuse Cases [Alexander, 2003].

These methods are considered as possible alternative methods that could be applied and the engineer has to select one.

For selecting a RE method, the engineer must take into account several characteristics of these methods and also some project characteristics in order to know whether a given method is suitable or not for the given project. The first group includes the guidance type (Criterion 1 – Cr_1), used approach (Criterion 2 – Cr_2), and formalism (Criterion 3 – Cr_3). The second group comprises the impact (Criterion 4 – Cr_4), level of innovation (Criterion 5 – Cr_5), and expertise (Criterion 6 – Cr_6). Thus, the engineer has six criteria that could guide him through the selection of the method to apply.

Information about the decision-making problem is summed up in the following decision matrix (Figure I.2.3.).

Criteria	NFR Framework	KAOS	Secure Tropos	GBRAM	Misuse Cases
Method Component Characteristics					
Guidance	predefined taxonomy	reuse of generic refinement patterns, heuristics	*No guidance*	documents analysis, heuristics	guidelines
Approach	explanatory	exploratory	systemic	*not applicable*	explanatory
Formalism	semi-formal	formal	formal	informal	informal
Project Characteristics					
Impact	3	1	3	1	2
Level of innovation	3	3	1	3	3
Expertise	normal	high	high	normal	low

Figure I.2.3. Decision Matrix for the RE Example.

Based on the project description, the engineer defines a rule for each criterion in order to express his preferences:

- *Impact on organisation:* maximize the value;
- *Level of innovation:* maximize the value;
- *Required expertise:* minimize the value;
- *Guidance:* a predefined taxonomy is better than heuristics, which is better than a simple guidelines;
- *Approach:* a systemic approach is better than an exploratory one, which is better than an explanatory one.

- *Formalism:* a formal approach is better than a semi-formal one, which is better than an informal one.

As we can see, there are two kinds of criteria: qualitative and quantitative. The impact and innovation level are quantitative as they are expressed by numeric values. Other criteria are qualitative. Sometimes, it is useful to express them through numeric values with respect of their order by choosing a scale (this kind of correspondence is called *utility function*). The selected scale has an impact on the final decision. For instance, the expertise could be expressed with a 3-degree scale. As this criterion must be minimized, and the other ones maximized, the quantified value will be represented as follows (Figure I.2.4.).

Criteria	NFR Framework	KAOS	Secure Tropos	GBRAM	Misuse Cases
Expertise	2	1	1	2	3

Figure I.2.4. Extract of the Decision Matrix with the Quantified Expertise Criterion.

This example allows introducing a decision-making situation composed of the main DM notions such as alternatives, criteria, and decision matrix containing the evaluations of the alternatives according to the given criteria. These elements constitute information available before applying a DM method. The proposal of the DM science is to provide guidelines for helping the engineer to apply a DM method in order to resolve the DM problem based on the available information, as in this case, to select a suitable RE method.

2.3.2. Basic Definitions

In this section, we introduce the main concepts of the decision-making science, such as decision problem, alternative, criteria, and evaluation matrix. A detailed description of these concepts could be found in [Pomerol *et al.,* 1993] [Roy, 1985] [Vincke, 1995].

2.3.2.1. Decision Problem

The decision *problem formulation* (or *problem*) [Roy, 2005] characterizes the result expected from DM. Relative to the potential alternatives (or actions), the engineer has to specify the terms he uses to expose the problem: Where does he focus his investigation? How does he conceive his prescription? To answer these questions, it is useful to situate the problem with regard to the four basic problem formulations.

The four basic problem formulations can be specified as follows [Roy, 1985]:

- ***P.α*** Clarify the decision through the choice of a subset, as restricted as possible, for the final choice of a single action. This subset should contain the "best" actions ("optimums") or, failing that, "satisfactory" actions.

- ***P.β*** Clarify the decision through a sorting consisting in an affectation of each action to a category, these categories being defined a priori (e.g. accepted, rejected, sent back for more information).
- ***P.γ*** Clarify the decision through a ranking obtained by grouping all or part of the actions into equivalence classes, these classes being ordered in a complete or partial way.
- ***P.δ*** Clarify the decision through a description in an appropriate language of the actions and their consequences.

Table I.2.1 sums up these problems.

Table I.2.1. Overview of the Four Basic Problem Formulations.

Problem	Description	Result	Procedure
P.α	Definition of a subset of potential alternatives (most often one alternative).	Choice	Selection
P.β	Affectation of the potential alternatives to some predefined clusters (categories).	Sorting	Affectation
P.γ	Organization of the potential alternatives into an ordered collection.	Ranking	Classification
P.δ	Description of the alternatives and their impact.	Description	Cognition

2.3.2.2. Decision Alternative (Action)

The concept of *alternative* designates the decision object. Any decision involves at least two alternatives that must be well identified.

[Roy, 1985] defines actions as follows as *"the representation of a possible contribution to the overall decision, likely, given the status of the decision process, to be considered independently and serve as point of application to decision support"*. This author differentiates action and alternative. The former leads to all possible solutions whereas an alternative is a particular type of action, which is characterized by the exclusivity of its application. In this thesis, we unify these two notions and define the alternative as a possible action available for decision-making. We use the distinction between global and fragmentary alternatives in order to show their exclusivity. An alternative is called "global" if it is exclusive from any other alternative inserted into the model; in the opposite case the alternative is called "fragmentary".

2.3.2.3. Decision Criteria

A *criterion* is a qualitative or quantitative expression of points of view, objectives, aptitudes or constraint allowing evaluating alternatives. It must be useful and reliable for the given problem. A criterion can be any type of information that enables the evaluation of alternatives and their comparison. There are many different kinds of criteria: intrinsic characteristics of artifacts or processes, stakeholders' opinions, potential consequences of alternatives etc.

[Roy 1985] defines a criterion as *"a function that associates each alternative with a number indicating its desirability according to consequences related to the same "point of view".* In this manner, the concept of criterion is clearly based on the notion of utility function *g* which provides two values *g(a)* et *g(b)* allowing to compare two alternatives *a* and *b* relative to the point of view underlying the definition of the criterion *g*. Based on this definition, a "preference rule" must be defined for a criterion. When dealing with criteria, the engineer must determine the wishful value for a criterion (for example, max. or min. for a numeric criterion).

2.3.2.4. Decision Matrix

We assume that the decision-maker is able to give to every alternative and for every criterion a numeric value or symbolic a_{ij} (i for the alternative, j for the criterion) which expresses an evaluation of A_i relatively to criteria j.

The matrix (a_{ij}) is called *decision matrix* or *performance table* (See Figure I.2.5.). Each row of this matrix expresses the performance of the alternative i relatively to the n considered criteria. Each column j expresses the evaluation of all alternatives relatively to the criterion j [Pomerol *et al.,* 1993].

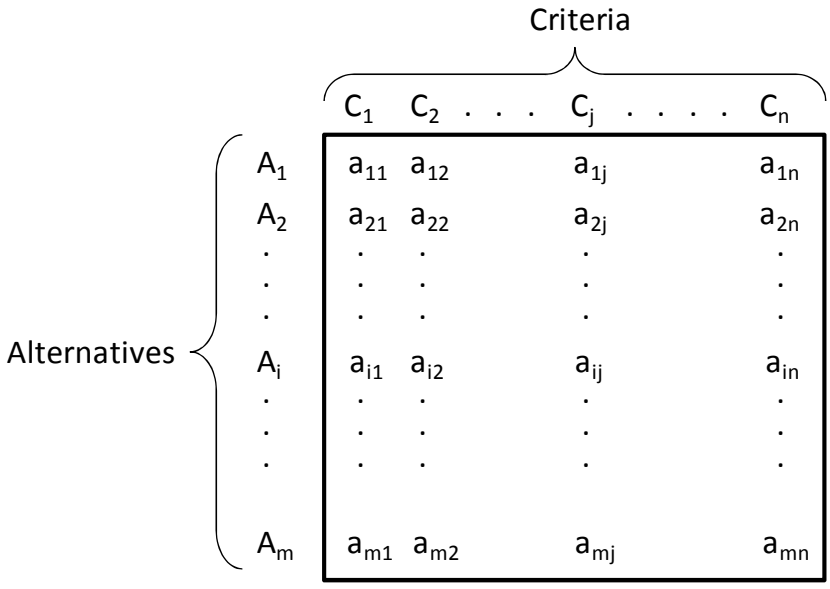


Figure I.2.5. Decision Matrix.

2.3.3. Mathematical Model

There are different mathematical models formalizing decisions. They consider different factors influencing the final decision. The decision-making problem may be formalized as follows:

A – alternatives' set presented for DM;

Y – result sub-set of alternatives *($Y \subseteq A$):*

a) selected alternatives' sub-set, in particular one alternative (choice problem),

b) ranked alternatives set (ranking problem), or

c) collection of alternatives sub-sets (sorting problem),

where a), b) and c) are the result set nature *(N_Y);*

δ – information that enables to compare the alternatives (in particular for MC problem – criteria set *(G)* and/or decision makers estimations *(E));*

π – selection rule (the steps to follow to bring out the sub-set *Y* from the *A*).

Given that the selection rule *(π)* depends on characteristics of *A, δ* and N_Y, the DM model can be presented in the following manner (1):

(1) $Y_i = \pi_i (A,\delta,N_Y)$, where *i* is the applied selection rule, $i \in [0;n]$

2.3.4. Decision-Making Actors

The first priority in a DM process is to assign actors. There are three main roles of actors participating in DM activities: stakeholder, IS engineer (when applied to the ISE field) and DM staff. Each typical actor has several roles in the DM process.

- A *Stakeholder* defines the decision problem, sets goals, expresses preferences on alternatives and criteria [Guidebook, 2001] and validates the final decision.
- An *IS engineer* evaluates alternatives and makes a proposal to DM stakeholders.
- *DM staff* is responsible for assisting stakeholders and IS engineers in all stages of the DM process [Guidebook, 2001]. DM staff includes a machine support for DM. In this case, this is a system actor. If actors are human, we call them decision-makers.

Decision-makers have the same roles as actors but have a complementary property, which is their type: individual or collective. A collective decision-maker is a group of decision-makers having the same goals and preferences and acting as a unique actor.

Actors contribute to the DM process at different stages. It is obvious that the same actor can play different roles in a specific DM process.

2.3.5. Decision-Making Process

Herbert Simon (1978 Nobel Prize in Economics) was the first to formalize the decision-making process. He suggested a model including three main phases: intelligence, design, and choice (the I.D.C. model) [Simon, 1960].

- ***I***ntelligence deals with investigating an environment for conditions that call for decisions.
- ***D***esign concerns inventing, developing, and analyzing possible decision alternatives.

- ***C***hoice calls for selecting an alternative from possible ones. This phase allows returning to the previous ones if the decision result is not satisfying.

This process was modified and extended in different ways. Currently, the commonly agreed and used decision-making steps [Guidebook, 2001] are defined as follows (See Figure I.2.6.):

1. Define a problem (necessity to define priorities),
2. Identify problem parameters (for instance, alternatives and criteria),
3. Establish evaluation matrix (estimate alternatives according to all criteria),
4. Select a method for decision making,
5. Aggregate evaluations (provide a final aggregated evaluation allowing decision).

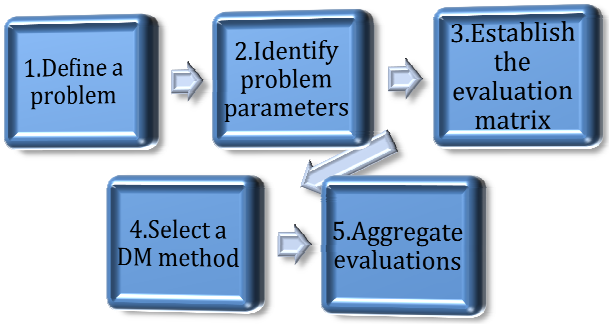


Figure I.2.6. Main Decision-Making Steps.

2.3.6. Decision-Making Methods

The goal of this section is to describe the most important DM methods. Based on [Bouyssous *et al.*, 2000] [Ballestero *et al.*, 1998] [Laaribi, 2000], we have established a list of the main DM methods as follows:

1. AHP (Analytic Hierarchy Process) [Saaty, 1980]
2. MAUT (Multi-Atribute Utility Theory) [Keeney *et al.*, 1993]
3. Outranking methods [Roy, 1996]
4. Weighting methods [Keeney, 1999]
5. Fuzzy methods [Fuller *et al.*, 1996]

These methods are detailed in turn in the following.

2.3.6.1. Analytic Hierarchy Process (AHP)

AHP, proposed by T.L. Saaty [Saaty, 1980], has generated a great number of methods. It is based on a dominance hierarchy, where the problem is decomposed into a hierarchy of levels. The law of hierarchical continuity requires that the elements of the bottom level of the hierarchy had to be comparable in a pair wise manner according to elements in the next level and so up to the focus of the hierarchy. Its principal steps are:

1. Structure alternatives in the form of a multilevel hierarchy including objectives, criteria and alternatives.
2. Carry out pair-wise comparisons of elements on each level; express the results of each pair-wise comparison into P_{ij} coefficient, which means that alternative i is more important than alternative j. The decision maker has a scale of reference (of "Equally Preferred = 1" to "Extremely Preferred = 9"). Moreover, $P_{ji} = 1/P_{ij}$. These coefficients are analyzed using a square matrix whose lines and columns relate to alternatives.
3. Normalize coefficients by calculating the sum of each column and dividing each matrix coefficient by its column sum.
4. Calculate the quantitative indicator (score) of each alternative "quality" (the score is given by the average of each row in the normalized comparison matrix) and the choice of the best alternative (which have the highest score value). Weighting reflects the criteria importance for the decision maker.

AHP is useful in situations where experts cannot give absolute estimations of alternatives. In this case, the method allows more exact comparison. The consistency of results is checked to be sure that decision-making is accurate. The disadvantages of AHP are essentially a limited number of alternatives to compare and the subjectivity of the quantification of the qualitative criteria.

2.3.6.2. Multi-Attribute Utility Theory (MAUT)

The Multi-Attribute Utility Theory (MAUT), proposed by H. Raiffa and R.L. Keeney [Keeney *et al.*, 1993], is a prescription for evaluating alternatives. According to MAUT, the overall evaluation of an alternative is defined as weighted addition or multiplication of its evaluations with respect to its relevant value dimensions.

The common denominator of all these dimensions is the utility for the evaluator. The simplified multiattribute utility function is as follows (2):

$$\text{(2)} \quad \text{Max E } [U(r_1, r_2 \ldots r_n)],$$

where Max E –maximal effect, U – utility function, r_i – attribute. Each attribute r_i is expressed as an utility function – the partial function. This method starts with the construction of partial functions for each attribute (criterion): r_i = fi (X). The multiattribute utility function (MAUF) represents either an addition, or a multiplication of the partial functions. MAUF has the following form (3):

$$\text{(3)} \quad U_k = \sum_{i=1}^{n} w_i u_i(r_k),$$

where:

- U - utility of the alternative K,

- w_i - weight of attribute I,
- $u_i(r_k)$ – attribute value I for the alternative K.

All alternatives are evaluated by using this function and the one that maximizes the utility is selected. Although the construction of the MAUF requires many time and efforts, the result makes it possible to analyze any alternative. However, the decision maker must fix all parameters in advance, it is not possible to change them if the analysis has began.

2.3.6.3. Outranking Methods

Outranking methods in [Bouyssou, 2001], [Kangas *et al.*, 2001], [Gomez-Limon et al., 2003] are inspired from the theory of social choice [Bouyssou, 2001]. The most known method is ELECTRE (ELimination Et Choix Traduisant la REalité, B. Roy / Elimination And Choice Corresponding to Reality). Outranking methods serve for approaching complex choice-based problems with multiple criteria and multiple participants. Outranking indicates the degree of dominance of one alternative over another. Outranking methods enable the utilization of incomplete value information and, for example, judgments on ordinal measurement scale.

It includes the following steps:

1. Calculation of the indices of concordance and discordance by couple of alternatives. These indices define the concordance and discordance following the assumption that alternative A is preferred to alternative B. The principle is that the decision maker estimates that alternative A is at least as good as B if the majority of the attributes confirm it (concordance principle) and the other attributes (minority) are not strong enough (discordance principle).

2. Definition of thresholds for the concordance and discordance indices. If the concordance index is higher than defined threshold and the discordance one is lower, then the first alternative is preferred to the other. If it is not the case, alternatives are incompatible (what means that A is preferred to B according to criterion X, and B is preferred to A according to the criterion Y).

3. Elimination of dominated alternatives. Then a first alternative subset is obtained, which can be either equivalent, or incompatible.

4. Iterative application of stages 2 and 3 with “lower” thresholds of concordance and discordance indices. A more restricted subset of alternatives is then elicited.

The procedure is applied until a suitable subset is obtained. The last subset includes the best alternatives. The order of the obtained subsets determines the alternatives’ scale according to their suitability to the given criteria.

An advantage of outranking methods is that they are based on step-by-step identification of decision makers' preferences. The detailed analysis makes it possible for decision makers to formulate their preferences and to define compromises between criteria. The incompatibility relation can be used to

find contradictory pairs of alternatives, to stop on a subset whose choice is justified (with available information). Difficulties can appear during the weight definition by the decision maker. Moreover, the appearance of the cycles (when alternative A is preferred to B, B is preferred to C and C is preferred to A) is rare but is not excluded.

2.3.6.4. Weighting methods

Weighting methods are based on the application of the addition or multiplication functions to the alternative values. These methods are characterized by the weight assignment to the decision criteria. Aggregation of the evaluations is based on the weighted sum or product. There are various interpretations of this principle. Several most known weighting methods are SMART (Simple Multiattribute Technical Rating), SWING, and Trade-off weighting [Keeney, 1999], [Mustajoki *et al.*, 2005] and [Poyhonen *et al.*, 2001].

The SMART method (proposed by W. Edwards), which appeared the first, includes the following stages: criteria scaling according to their importance, attribution of a value from 1 to 100 to each criterion, calculation of the relative importance of each criterion (the definition of criteria weights by importance analysis).

In the SWING weighting method (D. Winterfeldt and W. Edwards), all criteria are supposed bad. The expert chooses the one, which must be improved first and then a value of 100 is attributed to this criterion. The same operation is carried out with the other criteria to determine their values (by identifying criteria to be improved first).

In Trade-off weighting (H. Raiffa and R.L. Keeney), the decision maker compares two hypothetical alternatives according to two criteria; other criteria are invariable. The weights of these two criteria are refined so that the values of two given weighted alternatives have the same importance for the decision maker. This operation is repeated until all the weights are defined.

2.3.6.5. Fuzzy Methods

The fuzzy MC methods [Fuller *et al.*, 1996] and [Moisiadis, 2005] employ the fuzzy sets theory to add flexibility and to enrich methods by fuzzy parameters. The methods that are known as "traditional" ones, are refined by defining the decision criteria and/or attributes values like a set of fuzzy characteristics.

The fuzzy methods advantage is to take into account uncertainty and interdependence between criteria and alternatives. On the contrary, preferences determined by this type of method can be inexact and shall be discussed.

2.4. Decision-Making in Information System Engineering

In the science of Information Systems, actors are often faced with the problem of deciding how to proceed for selecting several actions or products. It can be which software to acquire, how to

prioritize requirements, select components, and so on. Decisions in IS Engineering (ISE) are motivated by:

- Time, effort, quality and resources constraints;
- Presence of multiple objectives and stakeholders;
- Necessity to deal with uncertain, incomplete and fuzzy information;
- Complex decision space.

We analyze in this section, the nature of ISE processes in order to show that all of them contain DM steps. Then, after a brief overview of DM tools, we give different classifications of decisions in ISE for clarifying situations in which a complex DM is justified.

2.4.1. DM Nature of the IS Engineering Processes

Process engineering is considered as a key issue [Rolland, 1998] by both the software engineering and information systems engineering communities. For instance, [Dowson, 1993] [Armenise *et al.,* 1993] [Jarke *et al.,* 1994] show that an improved development process will lead to improved productivity of the software systems industry and improved systems quality.

Traditional development processes have been conceived as a series of phases, each phase dominated by a particular activity organized into different sequences (e.g., “analysis,” “design,” “coding,” “testing” [Royce, 1970] [Boehm, 1988]. Later on, the specific activity sequences have been replaced by values and practices [Beck, 2005] [Abrahamsson *et al.,* 2002]. [Ralph *et al.,* 2008] highlights the fact that, in practice, the software developer is faced with a myriad of methodical choices as he can choose to execute one activity completely or partially, combine aspects of two or more different activities, or ignore another one. Decisions made through the process execution lead the IS engineer to enact a process which is unique and suited to the situation at hand.

However, decision points are often only addressed in a partial way. Most of the known processes lack methods to take into account the decision making inherent to any information system development process. According to Dowson [Dowson, 1988], process models can be classified into three groups of models called activity-oriented, product-oriented, and decision-oriented. [Rolland, 1998] refines this classification with a fourth group called context-oriented. Decision and context-oriented processes try to take decision-making into account as they integrate goals and arguments. Decision-oriented models are not only able to explain how the process is carried out but also why it is performed [Rolland, 1998] [Ralph *et al.,* 2008]. Context-oriented process models strongly couple the context of decision to the decision itself [Grosz *et al*., 1997] [Rolland *et al*., 2000]. It makes the notion of context (the coupling of the situation and the decision) central to process modelling. These two kinds of processes may be defined as *teleological processes*.

A teleological process is a process that takes into account the teleological behavior of the process execution. It describes intentions (goals, objectives) associated to a result that the designer wants to achieve. [Taylor, 1964] proposes a behavior classification as either goal-oriented (teleological) or stimulus-response behavior. [Malcolm, 1967] defines the teleological behavior as whenever the state

of a system and of its environment is such that a specific behavior is required for some event goal, then this behavior occurs. As a matter of fact, it is necessary to consider the possibility for the rules to change during the development process. [Cayla, 2008] emphasizes the link between this possibility to change and Veblen's teleological principles [Veblen, 1898], which integrate the agent's intentions in processes. This work defines the teleological process as 'dynamic equilibrium whose rules are determined endogenously, as a relation between the system's behavior and its intentions'. Such a teleological process may express changes within a process trajectory. For instance, after an engineer has decided a general route for his move, he will take the opportunity to revise his path during his progress, taking into account the information and knowledge he has gathered from his real environment.

[VandeVen *et al.,* 1995] defines the teleological processes as describing the behavior of agents making actions to reach a purpose or a goal (an envisioned end-state). [Ralph *et al.,* 2008] highlights that this is consistent with software development as: the teleological agents are the software development team and the project manager; the end-state represents the software product; the development team performs actions (coding, testing…) that correspond to the teleological steps, and the project manager monitors performance. The notion of purpose is also essential for any organization because it meets several objectives. Similarly, the information system is created because there are several organization goals to be met and the system's functions and properties are defined by the organization's goals that the system intends to fulfill [Loucopoulos *et al.*, 1995].

From the teleological viewpoint, any IS engineering process contains steps of two kinds: decision-making and decision performing steps. DM steps alternate with decision performing steps. On the one hand, the teleological reality of processes' existence requires decisions. On the other hand, engineering-related decisions result from the need to satisfy practical constraints such as quality, cost or performance [Ruhe, 2003]. Although these aspects are important, the arguments to carry out final decisions are actually poor. Choices are made in an intuitive and hazardous way [Ruhe, 2003] [NgoThe *et al.,* 005].

One could claim that this situation is due to the lack of DM methods, or to the fact that the methods defined in the operational research area are not adequate to the kinds of problems faced by ISE. However, the state of the art [Roy, 2005] [Kornyshova *et al.,* 2007] shows that there are numerous methods, from very complex ones to very simple ones. These methods can be used in many different DM situations (for instance, in IS engineering) [Boonstra, 2008]. With regard to ISE methodologies, the issue of DM was already explored with respect to requirements engineering [Rolland *et al.,* 2000] [NgoThe *et al.,* 2005], to method engineering [Aydin, 2006], and, to systems engineering [Ruhe, 2003]. Several examples of MC methods application can also be mentioned: AHP for prioritizing requirements [Ruhe *et al.*, 2003] [Karlsson *et al.*, 1997]; Saeki uses weighting method to deal with software metrics [Saeki, 2003]. Some researches deal with DM issues in the IS engineering domain. For instance, [Ruhe, 2003] emphasizes fundamental principles and expectations on software engineering decision support. Despite these examples, we have established that no attempt has been conducted to systematically guide DM in various situations within the IS engineering context and this will be part of our research in this thesis.

2.4.2. DM Tools Usage in IS Engineering

With regard to tools, we have distinguished four types of DM implementation:

- Tools that implement different DM methods (for instance, different tools supporting outranking methods are described in [Pomerol *et al.*, 1993]). Each tool deals with a specific MC method.
- Tools enabling selection of appropriate DM method [Ulengin *et al.*, 2000] [Hanne, 1999] [Ozernoy, 1996]. A tool guiding selection of adapted DM method in the specific field of IS is not provided actually. Furthermore, there is no tool that allows integrating DM methods with IS.
- DM functionalities embedded into tools (for example, OvarWeb[2]). This kind of tools usually uses the weighting method to make decisions.
- The Decision Deck project [Decision Deck, 2009] is more similar to our work. Within this project, a platform is developed to deal with different DM methods and to provide DM decision aid. Nevertheless, Decision Desk does not include the selection of DM methods.

2.4.3. Typology of Decisions in IS Engineering

Classifying decisions allows us to investigate their specific features and, consequently, to find more efficient solutions to problem solving taking into account the variety of conditions in which decisions are made. However, given the complexity of these conditions (or factors impacting decisions), together with stakeholders' goals and decision structure, a simple typology could not be constructed. This explain the different existing classifications [Lukitcheva *et al*., 2009] [Shemetov *et al*., 2009].

Applied to the IS engineering field, we propose the following classification of decisions:

1. Nature of the decision-making process;
2. Number of alternatives;
3. Frequency of making similar decisions;
4. Time of the consequences occurrence;
5. Number of actors impacting decisions;
6. Importance degree of the time constraint;
7. Complexity of decisions;
8. Consideration of the evolving DM conditions;
9. Goals number;
10. Goal time dependency;
11. Risk degree.

These classifications are detailed in the following.

[2] http://www.atpiware.fr/prod_ovar.htm, accessed by February, 2009.

1. Nature of the decision-making process

According to the nature of the decision-making process, decisions are classified as follows:

- *Intuitive* – Decision based only on feelings. In this case, a decision maker does not compare the pros and cons of each alternative and does not need a good comprehension of the situation.
- *Expertise-based* – Decision is based on acquired knowledge and experience. A decision maker uses knowledge about similar situations already occurred for forecasting results of the decision. Often he mades decision by analogy with a successful one of the past.
- *Rational* – Rational decisions are made with aid of an analytic process or DM methods.

2. Number of alternatives

Concerning the number of alternatives, decisions can be binary, multi-alternative and innovative.

- *Binary* – This is a choice between two opposite solutions. It can be, for instance, to buy or not to buy a tool, to acquire new software or to develop a homemade solution.
- *Multi-alternative* – This decision is characterized by the presence of multiple alternatives.
- *Innovative* – This occurs when alternatives are not clearly defined. In this case, the preparation of the decision takes more time due to the definition of the alternatives set.

3. Frequency of making similar decisions

The possible frequency of making similar decisions is:

- *One-time* – The decision is made only one time and is not repeated later.
- *Cyclic* – Such decisions have a known-cycle and are repeated regularly, for instance, yearly decisions.
- *Frequent* – Decisions are related to problems that appear often but without a predefined cycle. They can be considered as decisions made continuously.

4. Time of the consequences occurrence

Regarding to the time of the consequences occurrence, decisions are classified in the following manner:

- *Strategic* – Decisions that aim at attaining the strategic goals of a company.
- *Tactic* – Decisions that have a medium-term impact.
- *Current* – Current decisions are short-term ones.
- *Operational* – Decisions that deal with day-to-day activities.

5. Number of actors impacting decisions

Number of actors impacting decisions can be:

- *Determinative* – Decisions made by only one decision maker.
- *Competitive* – Decisions made by two decision makers.
- *Adaptive* – Decisions made collectively by several decision makers.

6. Importance degree of time constraint

According to the degree of importance of time constraints, decisions are:

- *Real-time* – A decision made quickly for controlling a current process.
- *One-step* – Decisions made within a predefined time slot.
- *Without time constraint* – Decisions that do not have any constraint on their formulation and execution.

7. Complexity of decisions

The complexity of decisions is:

- *Simple* – Decision made during execution of an activity.
- *Complex* – Decision impacting several activities related to each other.

8. Consideration of the changing DM conditions

With regards to the consideration of the changing DM conditions, decisions can be:

- *Flexible* – Decisions that are based on processes allowing to take into account changes of the DM conditions.
- *Rigid* – Rigid decisions have only one variant of execution even if conditions change.

9. Goals number

The goal number can be:

- *Small* – Decisions made for satisfying a small number of goals.
- *Large* – Decisions taking into account a large number of goals.

10. Time dependency of the goals and conditions

Related to the time dependency of the goals and DM conditions, decisions can correspond to:

- *Static goals* – Goals and DM conditions are relatively stable in time.
- *Dynamic goals* - Goals and DM conditions vary through the time.

11. Risk degree

According to the risk degree, decisions can have:

- *Acceptable risk* – Decisions are made in the condition of the acceptable risk. The existing risk is not significant enough to impact decisions.
- *Critical risk* – Decisions are made with a critical degree of risk.
- *Disastrous risk* – Decisions must take into account the risks that have an important impact.

This typology may not be complete because of the diversity of important factors impacting decisions. However, the presented list of decision types offers a detailed vision of the variety of decisions in ISE and, consequently, the complexity of DM situations.

In addition, it allows understanding whether a simple DM method is sufficient or a scientific DM method is required to resolve the problem. For this reason, we have positioned the possible types of decisions on a scale. Figure I.2.7 shows the progression of the need for method-based decision-making depending on the decision types described above.

Nature of the decision-making process	Intuitive	Expertise-based	Rational	
Number of alternatives	Binary	Multi-alternatives	Innovative	
Frequency of making similar decisions	One-time	Cyclical	Frequent	
Time of the consequences occurrence	Operational	Current	Tactic	Strategic
Number of actors, impacting decisions	Determinative	Competitive	Adaptive	
Importance degree of the time constraint	Without time constraint	One-step	Real-time	
Complexity of decisions	Simple	Complex		
Consideration of the changing DM conditions	Rigid	Flexible		
Goals number	Small	Great		
Goal time dependency	Static goals	Dynamic goals		
Risk degree	Acceptable risk	Critical risk	Disastrous risk	

Figure I.2.7. Needs for Method-Based Decision-Making depending of Decision Types in ISE.

This representation offers an overview of the situations when complex and time-consuming decision-making is justified and when sophisticated DM is not necessary. For instance, strategic decisions must be made using a suitable DM method.

2.5. Decision-Making Method Selection

DM methods are often selected arbitrarily: sometimes the analyst is already familiar with a procedure, other times a method is developed in an ad-hoc manner, it also happens that a method is chosen simply because a tool that supports it is available [Hanne, 1999] [Laaribi, 2000] [Ulengin *et al.*, 2000]. Experience also shows that there is no DM method that is able to deal with all decision-making problems [Ballestero *et al.*, 1998] [Laaribi, 2000]. Many methods are proposed to deal with a specific kind of decisions. For instance, [Papadacci, 2008] presents a method for prioritizing requirements. In fact, each situation demands a specific DM method. The impact of the choice of a method on actual decisions is also well known, as well as the consequences of poor decisions. Several researchers propose state-of-the art descriptions of DM methods, including Multi-Criteria Decision-Making (MCDM) methods. However, to our knowledge, a well structured state-of-the-art related to the selection of DM methods is missing.

Two main groups of DM methods are single and multiple DM. The first step consists in selecting one or more criteria. This selection is rather simple when a unique criterion is suitable in the situation and when only one factor is available allowing comparing alternatives. Multiple DM is richer, and multi-criteria DM methods are numerous. For this reason, we focus our attention in this section[3], on the selection of an appropriate MCDM method.

The Multi-Criteria Decision-Making literature reveals a large number of methods ranging from outranking methods, to analytic hierarchy process, multiattribute utility theory, weighting methods, fuzzy methods, or multiobjective programming (see, for example, [Bouyssous *et al*., 2000] [Roy, 2005]). Considering the diversity of these methods, few attempts have been made to guide their selection.

Eight selection approaches [Ballestero *et al.*, 1998], [Felix, 1995], [Hanne, 1999], [Laaribi, 2000], [Olson *et al.*, 2000] [Ozernoy, 1996], [Ulengin *et al.*, 2000] and [Vincke, 1995] were considered to be discussed in this thesis. Based on the study of these approaches, we concluded that the selection approaches focus on different MC methods, the comparison criteria are not the same, and the approaches for comparison are not the same. In order to improve comprehension, we set two goals: (a) to emphasize similarities and differences of selection approaches and (b) to identify the requirements for a 'good' MCDM selection approach.

To achieve this, our research strategy was to develop a structured analysis framework that guides the comparison of MCDM selection approaches. A first review of the eight selected approaches revealed a number of attributes that could be gathered into different facets that formed two particular views.

[3] In the following, the term "method" is used to designate DM methods; whereas the term "selection approach" relates to the techniques used for choosing a DM method in a given situation.

The first view concerns characteristics of the selection approaches. The second view is about MCDM methods' characteristics that are addressed for selection. In the first view, the analysis framework defines the properties of selection approaches themselves. In the second view, the selection of a method must be achieved by taking the situation at hand [Ballestero *et al*., 1998] [Laaribi, 2000] [Vincke, 1995] and method's features [Ulengin *et al*., 2000] into consideration.

In the following, we present our evaluation framework, an overview of the eight selection approaches, our comparative analysis, and the list of requirements for a "good" selection approach.

2.5.1. Evaluation Framework for Selecting Multicriteria Methods

The selection approaches differ by their specific characteristics and by their ability to take into account different properties of MCDM methods. We have defined an evaluation framework according to the in-depth analysis of the selection approaches characteristics, and of the properties of the selected MCDM methods. For this reason, we propose to analyse MCDM methods' selection approaches according to two views:

- the first view deals with the characteristics of the selection approaches themselves;
- the second one deals with the characteristics of the MCDM methods.

Each view comprises facets that facilitate the study of the selection approaches. Each facet includes a set of attributes (characteristics). The two views are respectively developed in the following sub-sections.

2.5.1.1. The "Characteristics of Selection Approaches" View

The analysis of the eight MCDM methods' selection approaches has revealed a number of characteristic attributes that we gathered under two facets: features and context.

The **feature facet** deals with the attributes that characterize the selection approaches themselves: objective, comparison approach, structured algorithm, nature of the selection approach, or capitalization.

- The *approach objective* concerns the goal that was adopted by the authors for MCDM methods comparison. Initially, the goal is a method selection. Nevertheless, several authors suggest comparing the MCDM methods in order to improve understanding and practical use. All these approaches are interesting due to their analysis of the MCDM methods.
- The *basic approach used to compare MCDM methods* can be defined as the mechanism that allows to select and to analyse them. It reflects the degree of accessibility of the approach by users. Some approaches are supposed to be applied by users themselves; other must exploit a tool or a third person. The approach, which is easy to apply, is more preferable in practice because it is not necessary to buy a tool. The basic approaches found through this state-of the-art are tree analysis, distance function model, drawing up properties lists, expert system, neural network, decision-making steps analysis and framework application.

- The presented approaches differ from each other by the *presence of a structured algorithm*. Some of them suggest a mechanism to carry out the selection. Others only rely on a verbose comparison of the selected methods. A structured algorithm guides users more systematically and pro-actively. On the other hand, textual approaches have the advantage of being easily adoptable and adaptable in practice when they are simple.
- Some authors such as [Felix, 1995] envisage the problem of MCDM method selection as a multicriteria problem itself. In this case, the *nature of the selection approach* is multicriteria based. Other researchers [Laaribi, 2000] and [Ulengin *et al*., 2000] consider that this understanding generates a “vicious circle”. They usually accept that it can be taken as a MC problem, but they are against using a particular MC method for selection. In this case, the nature of the selection approach is not multicriteria.
- Often, the same problem appears repeatedly from one decision cycle to another. In this case, the decision-making may involve two approaches: restart all procedure from scratch to obtain a new result, or adapt the previous solution to the new conditions. Adaptation is possible thanks to *capitalization*. The approaches may or may not contain a possibility to capitalize on the selection results.

The **context facet** corresponds to attributes that deal with the context of the MCDM method selection: application domain, considered DM methods, problem specification, DM steps taken into account, or tool.

- Some selection approaches were created to be used in specific domains while others were generalized to suit any domain. Consequently, the approach *application domain* can be defined as generic or specific. The generic approach is, of course, interesting because of its adaptation to any context of MCDM method selection. However, specific approaches have the advantage to be well fitted to the given decision problem.
- Different selection approaches consider different collections of *MCDM methods*. Some authors suggest comparing all major groups, others only one or several groups.
- The characteristic of *problem specification* indicates if the authors have mentioned that MCDM method selection depends on problem situation and have specified its characteristics. In fact, not all approaches intend to select a method depending on a specific problem. The possible values of these attributes are: "no" – then the approach does not take into account the problem specificity; "yes, not specified" – then the approach indicates the necessity to analyse the problem specificity but does not propose a typology of problem characteristics; and "yes, specified" – then the approach includes such typology.
- All MCDM methods are employed in the context of a process that involves decision-making. In order to understand the role of MC methods, their contribution on different *decision-making steps* must be highlighted.
- The presence of a *tool* facilitates the adoption of the MCDM methods selection approaches. However, tools can be costly and are not always adapted to the specific problem situation. Besides, purchasing a tool is only interesting when intended to be used many times.

2.5.1.2. The "Characteristics of MCDM Methods" View

Three basic concepts are generally used to define multicriteria problems [Roy, 2005]: the problem, alternatives (potential actions), and criteria collections. Given that our state of the art deals with MCDM method selection, we believe a fourth concept is needed to characterize methods in the context of their selection, namely the usage in practice. These concepts form four facets: problem, potential actions, criteria and MCDM method usage. Each selection approach is analyzed in order to show if it takes into account the characteristics of MCDM methods. Therefore, the possible values for these characteristics are yes or no.

In the **problem facet** two kinds of attributes can be used to characterize a MCDM problem, namely the type of decision problematic and the problem scale.

- As mentioned above, the *type of decision problematic* [Roy, 2005] can be defined by the result expected from an application of the selected MCDM method. Each decision-making method is able to support a specific type of decision (*choice, sorting, or ranking*), therefore take into account when selecting the appropriate decision-making method.
- The *problem scale* characterizes MCDM methods according to the size of the problem with which they are able to deal. For example, in the context of an organisation the problem to consider can concern a workplace, a department, the enterprise or its corporation as a whole. Of course, the amount of resources and of organizational efforts needed to deal with the decision problem will be different in each case.

A set of potential actions may vary from one situation to another. We propose to use the following attributes in order to characterize the **potential actions facet**: number of alternatives, ability to consider new alternatives, incompatibility and conflicts, organization of the alternatives, and nature of the alternatives set.

- The selection of methods depends primarily on the *number of alternatives*. In fact, several MCDM methods are not adapted to a large number of alternatives. In addition, the number of alternatives is crucial when choosing a method accompanied by a tool: it is more reasonable to purchase a tool when the number of alternatives is large.
- The need to be *able to consider new alternatives* results from the fact that the potential actions collection is not stable and may evolve from one moment to another, in particular when decisions need to be revised. New potential actions may appear, several potential actions disappear, and others change their properties. In practice, cases of repetitive and cyclic problems are frequent. MCDM methods handling the possibility to deal with repetitive problems facilitate the decision-making.
- In some situations, considered alternatives have interconnections, incompatibility and conflicts. Therefore, the chosen MCDM method must take into account *incompatibility and conflict of alternatives*.
- Potential actions may form a hierarchy. In this case, the approach of MCDM methods selection must take into account *alternatives' organization.*
- Lastly, the *set of alternatives* could have a continuous or discrete *nature*.

Within the **criteria facet**, MCDM methods selection approaches characterize decision criteria following the four attributes: data type, measure scale, criteria weighting, and criteria interaction.

- There are two kinds of *data types*: quantitative and qualitative. Certain methods deal with two kinds of data type (e.g. outranking methods) while others require a quantification of qualitative values (e.g. weighting methods) that deforms the initial information.
- Criteria must be measured. A *measure scale* therefore characterizes them. Types of scales depend on the nature of the relationship between criteria values. These are nominal, ordinal (restricted or unrestricted), interval, ratio, and absolute. B. Roy indicates that two kinds of scales are more consistently used in MCDM: nominal and ordinal [Roy, 2005]. Some methods take into account criteria with different measure scale; others do not.
- Several MCDM methods comprise *criteria weighting*, others do not. This aspect must be taken into account for selecting a method.
- The criteria may be independent, cooperative, and conflicting. For this reason, it is important to analyze possible *interactions between criteria*.

Five attributes were gathered under the **usage facet** to characterize the intended context of use of the MCDM method while a decision maker undertakes selection. These are tool, approach for giving partial and final evaluations, easiness of use, cost for implementing and decision maker preferences.

- The presence of a *tool* is an important selection criterion for practitioners who are concerned with rapid application of selected methods.
- *Approaches for giving partial and final evaluations*. Partial evaluations are estimates of potential actions corresponding to each criterion and final estimations present a synthesis of the partial evaluations. Partial evaluations are, for example, simple measuring or pair wise comparison. Final estimations can be carried out for example, by outranking or summing up.
- Characteristic "*easiness of use*" is more difficult to deal with as it is vague and can cover different aspects such as easiness of understanding, rapidity of appropriation by users or even easiness of implementation in a house-made software tool. This is again an important characteristic for decision makers, which have different preferences regarding the difficulty of using MCDM method.
- Another important characteristic is *costs for implementing* a method [Hanne, 1999]. Costs include costs for implementing method, for purchasing tool, and for training decision makers.
- *Decision maker preferences* include him (her) understanding of different methods, their skills to use these methods, and habits to use a given MCDM method.

2.5.2. Overview of the Reviewed Selection Approaches

The objective of this section is to present a state-of-the-art of the approaches allowing multicriteria methods selection. There is a limited number of selection approaches. For this reason, in our analysis, we used the approaches aimed not only to select one method but also to simply compare them with regard to various criteria. The approaches adopted for the comparison are:

1. Laaribi's approach [Laaribi, 2000],
2. Hanne's approach [Hanne, 1999],
3. Ulengin et al.'s approach [Ulengin *et al*., 2000],
4. Vincke's approach [Vincke, 1995],
5. Felix's approach [Felix, 1995],
6. Ozernoy's approach [Ozernoy, 1996],
7. Olson et al.'s approach [Olson *et al*., 2000],
8. Ballestero and Romero's approach [Ballestero *et al*., 1998].

2.5.2.1. Laaribi's approach.

The central element of Laaribi's approach [Laaribi, 2000] is a "correspondence frame" allowing establishing links between characteristics of problem and characteristics of MCAP (Multiple Criteria Aggregation Procedure). To carry out the analysis of correspondence frame, the author proceeds according to the following steps.

The first step is the identification of the problem characteristics. Detailed analysis of all the aspects of the decisional problem is carried out in order to identify the vector of these characteristics. The next step is the identification of appropriate characteristics of MCAP in order to establish the correspondence between characteristics of decisional problem and conditions of MCAP usage. The result is the obtention of characteristics of MCAP, which are adapted to the decisional problem at hand.

Then, to select an appropriate MCAP the author suggests the following two steps:

1. Initially, a subset of MCAP is chosen. The author proposes a tree structure to choose a group of methods. By going up the branches of the tree according to the problem characteristics, a category of methods is selected. The tree has two branches: discrete and continue methods. In this manner, a restricted set of MCAP is identified (while passing from a node to another according to characteristics of appropriate decisional problem).
2. Then, a particular method is selected by taking into account MCAP characteristics. To choose a particular method the author considers the characteristics related to the MCDM methods use in practice: easy to understand by users, inspiring the confidence of the user, easy to program and having a data-processing support.

The MCDM methods which have been compared are the following: discrete methods: MAUT, Outranking, AHP; continue methods: interactive methods, goal programming.

To compare these methods, the author proposes to analyse types of decision problematic, nature of actions set (finite, infinite), information nature to obtain on criteria (information nature: quantitative, qualitative; intra-criteria information: value function (utility function or distance function), criterion discriminatory power (thresholds and orders); inter-criteria information: relations between criteria), result of evaluation (result type: specific, distributional; inaccuracy of information: the method tolerates it or not).

This approach was developed for geographical information systems (GIS). It proposes a detailed algorithm of MCDM method selection that is valid for different domains. It establishes links between characteristics of the problem and those of methods. Characteristics taken into account are very different. However, the arborescence analysis relates only to the first stage of selection. As for second stage, the author does not suggest a structured algorithm neither to analyse characteristics (such as the MCDM methods use in practice), nor to carry out their choice.

2.5.2.2. Hanne's approach

Hanne [Hanne, 1999] considers the problem of MCDM methods selection as a meta-decision problem. He shows that this problem might be solved in two ways: by selecting one MCDM method from a finite set of methods or by parameters assessment.

In the first case, the author generalizes the criteria usually used for MCDM method selection. The criteria for MCDM methods evaluation and selection form four groups: suitability for the problem type, criteria based on solution concepts, implementation-oriented criteria, and criteria depending on the specific decision situation. He positions different MCDM method families according to these criteria groups.

In the second case, the author suggests an approach for designing MCDM method. This approach is based on "parameter assessment". The parameter is additional information that serves to adapt MCDM method to the situation at hand. Examples of parameters are weights, achievement levels, threshold values, trade-offs, etc. A parameter optimization model permits the MCDM method design by choosing parameters from a continuous set.

The author mentions some other approaches of MCDM methods selection. Furthermore, he suggests a possibility to use machine learning in order to resolve the meta-decision problem for repetitive decision situations.

2.5.2.3. Ulengin et al.'s approach

In [Ulengin *et al*., 2000], the authors propose a framework of "Integrated DEcision Aid model enriched by Artificial Neural Network ($IDEA_{ANN}$)". The similarity between the characteristics of the methods and the basic parameters of the decisional situation is analyzed using ANN.

The approach consists of three steps:

1. *Structuring and modelling of the problem*. The decision maker constructs cognitive maps, detects loops (with a hierarchical presentation), and chooses whether he takes into account all the hierarchy or just fundamental objectives. Then, $IDEA_{ANN}$ seizes the characteristics of the problems: type of decision problem (choice, classification, rating); size of the problem (small and broad number of criteria and alternate); technical selection of the preference by DM (direct rating, trade-off, pair wise comparison); DM's preference structure (according to the presence of the thresholds and type of order (partial or complete); necessity for uses of relative

importance (presence of weights); and nature of performance values (quantitative, qualitative).

2. *Matching of the decisional situation with appropriate MCDM methods*. Initially, a suitable cluster of methods is selected. Decision maker chooses methods, which correspond exactly to the six characteristics of the situation of decision-making using the artificial neural network.
3. *Selecting a concrete method within a cluster* (a similar procedure is used).

Four groups of discrete methods are present in this approach: elementary methods, interactive methods, value based methods, outranking methods

The method proposes guidance throughout the decision-making process and is supported by a software tool. However, it deals y with discrete methods, only.

2.5.2.4. Vincke's approach

The author [Vincke, 1995] suggests a method of comparing decision-aid methods in order to improve their understanding and to select one.

The proposed methodology consists in defining a list of properties that should be respected and in verifying which of them are satisfied by the compared methods. The list of properties is compiled for better understanding of the methods.

This methodology includes the following steps:

1. Establishing a list of properties.
2. Alternative methods are analysed in order to understand which properties they satisfy and which they don't.
3. The method that dominates other methods is chosen (for example, we have two methods: if one method satisfies all the properties and the other satisfies some properties and violates others).

The author remarks that it is often possible to not have dominance between alternatives but he does not propose a solution to deal with such cases.

This approach is illustrated by a small example with two variants of the ELECTRE II method. However, it has a generic nature and may be applied to all MCDM methods.

The author indicates the necessity of analysing the problem situation in order to select one of the MCDM methods; however, he does not give a typology of problem characteristics.

2.5.2.5. Felix's approach

Felix's approach [Felix, 1995] aims at analyzing MCDM methods that take into account goals for the alternatives in decision-making process.

Therefore, the author compares two MCDM methods (AHP and fuzzy Decision Making based on Relationships between Goals – DMRG) according to four criteria: representation of decision alternatives, representation of decision goals, role of decision priorities, and way of aggregation.

The author indicates that the choice of a MCDM method depends on a decision situation. He uses a small number of criteria that are generic.

2.5.2.6. Ozernoy's approach

The approach [Ozernoy, 1996] is based on Expert Systems. The authors do not explain the basis of their methodology. Therefore, in this article they consider experimental comparisons and Expert Systems. The most important solution made by these researchers is that the "best" MCDM method for resolving all problems does not exist.

The purpose of this work is to clarify the complexity of the selection problem, and to facilitate understanding, evaluation, and improving decision-making.

The author justifies the need for a systematic, logical and justifiable approach. He suggests descriptions of experimental comparisons of MCDM methods and MCDM Expert Systems. He proposes an expert system based tool in order to select a MCDM method corresponding to the problem situation. However, the underlying methodology is not explained and problem characteristics are not specified.

2.5.2.7. Olson et al.'s approach

The approach presented in [Olson *et al.*, 2000] aims at defining the role of the MCDM in decision-making steps.

The importance of alternatives according to the criteria can vary in different decision-making steps (at the beginning and the end of the analysis). That is why the authors suggest analyzing dynamic parameters of decision-making that must be considered in the development of decision-making aid. Dynamic components during the analysis of a multicriteria problem are changes in criteria importance and alternatives changes.

The authors advise application of different MCDM methods (more precisely of MAUT, AHP, Outranking, Preemptive, and Preference cones) depending on decision-making steps. The criteria are tools and uses, strategy, weight elicitation, and score elicitation.

This approach is interesting because of its dynamic parameters analysis and the relation with decision-making steps.

2.5.2.8. Ballestero and Romero's approach

The approach [Ballestero *et al.*, 1998] studies relations between MCDM methods based on distance function. The authors aim at creating technical and analytical links between various MCDM methods.

They use a mathematical apparatus, more exactly – the distance function model, which allows to look for a common root.

Using mathematical operations, they obtain models of the following methods: traditional mathematical programming, weighted goal programming, lexicographic goal programming, min-max goal programming and compromise programming.

The authors recognize that the relative advantages and disadvantages of MCDM methods depend on the characteristics of the problem situation. Nevertheless, these kinds of characteristics are not presented in [Olson *et al.*, 2000].

2.5.3. Comparative Analysis with the Evaluation Framework

This section offers a comparative analysis of the eight approaches presented above based on our framework.

2.5.3.1. Characterization of the Selection Approaches

Table I.2.2. summarizes our analysis. The rest of this section comments on approaches facet by facet.

The majority of approaches [Hanne, 1999] [Laaribi, 2000] [Ozernoy, 1996] [Ulengin *et al*., 2000] and [Vincke, 1995] *aim a*t selecting a method. Furthermore, Hanne suggests a method design; Vincke means a better understanding of MCDM methods; Felix proposes a simple comparison.

All approaches use different *basis to compare MCDM methods*. Let us mention only, that some of approaches have a simple theoretical basis and could be applied easily in practice (for example, arborescence analysis, properties list complying). Others are based on more complex concepts: expert systems or neural networks that usually require a tool to be applied.

Several *structured algorithms* for selecting a MCDM method are available, in Laaribi [Laaribi, 2000], Ulengin et al. [Ulengin *et al.*, 2000], and Vincke [Vincke, 1995]. Hanne [Hanne, 1999] presents a structured algorithm for design MCDM method. In other approaches, the authors suggest simple comparisons of MCDM methods. It may guide users in MCDM method selection, but their choice is intuitive.

The *selection approach nature* is multicriteria in [Felix, 1995] and [Hanne, 1999]. The other researchers do not accept this viewpoint.

One approach (Ulengin et al. [Ulengin *et al.*, 2000]) allows the *capitalization* of selection results thanks to the use of neural networks. Hanne's approach suggests the reuse of selection results (machine learning) based on neural networks.

Table I.2.2. Comparative Analysis of the MCDM Methods Selection Approaches according to their Characteristics.

Characteristics	Laaribi	Hanne	Ulengin et al.	Vincke	Felix	Ozernoy	Olson et al.	Ballestero/ Romero
Features								
Approach's objective	Selection	Selection, design	Selection	Selection, understanding	Comparison	Selection	MCDM on DM steps	Links between methods
Basic approach used to compare MCDM methods	Arbore-scence	Textual	Neural Network	Properties list complying	Textual	Expert System	DM steps analysis	Distance function
Presence of structured algorithm	Approach	Approach for design	Approach	Approach	Comparison	Comparison	Comparison	Comparison
Selection approach nature	Not MC	MC	Not MC	Not MC	Not MC	MC	Not MC	Not MC
Capitalization	No	Yes	Yes	No	No	No	No	No
Context								
Application domain	Specific (GIS)	General	General	General	General	General	General	Specific (Economy)
Problem specification	Yes, specified	Yes, specified	Yes, specified	Yes, not specified	Yes, specified	No	Yes, not specified	No
Decision making steps	No	No	Yes	No	No	No	Yes	No
Tool	No	No	Yes	No	No	Yes	No	No

There are two approaches [Ozernoy, 1996] [Ulengin *et al.*, 2000], which offer *tools* in order to carry out the selection. The first is based on neural network and the second on expert systems.

Application domain is general in all approaches except the Laaribi's [Laaribi, 2000] and Ballestero and Romero's [Ballestero *et al.*, 1998] ones. Nevertheless, they may be applied in other areas with some considerations. For example, in [Laaribi, 2000], the list of problem characteristics has to be revised.

We present the *MCDM methods* considered by different selection approaches in Table I.2.3. The approaches of Ulengin et al. [Ulengin *et al.*, 2000], of Ozernoy [Ozernoy, 1996], and of Ballestero and Romero [Ballestero *et al.*, 1998] have a limited number of methods that they consider. The others compare a greater number of methods.

Table I.2.3. MCDM Methods Compared by Selection Approaches.

MCDM Method	Laaribi	Hanne	Ulengin et al.	Vincke	Felix	Ozernoy	Olson et al.	Ballestero and Romero
Outranking methods	+	+	+	+	-	+	+	-
AHP	+	+	+	-	+	+	+	-
MAUT	+	+	-	-	-	+	+	-
Weighting methods	+	+	+	-	-	+	-	-
Fuzzy methods	-	+	-	-	+	+	-	-
Multiobjective programming	+	+	-	-	-	-	-	+
Others	+	+	+	-	-	-	+	+

The three first approaches [Hanne, 1999] [Laaribi, 2000] [Ulengin *et al*., 2000] and Felix's approach specify *problem characteristics*. Two approaches [Ballestero *et al.*, 1998] [Olson *et al*., 2000] foresee a problem specification but do not propose a typology. The authors of [Ballestero *et al.*, 1998] and [Ozernoy, 1996] do not consider this issue.

2.5.3.2. Characterization of the MCDM Methods

Table I.2.4 offers an overview of the eight approaches according to second view of the proposed framework.

Only three approaches [Hanne, 1999], [Laaribi, 2000], and [Ulengin *et al*., 2000] take into account the type of result that must be obtained following the method application. In [Laaribi, 2000], the author suggests the use of *decision problematic* in two cases: in order to describe a problem situation and in order to characterize a MCDM method. A method will be chosen if it corresponds to the type of decision problematic required in a given situation. In [Hanne, 1999] this characteristic is described as the "desired solution concept". For [Ulengin *et al*., 2000], it is a problem feature.

Table I.2.4. Comparative Analysis of the MCDM Methods Selection Approaches according to the Characteristics of MCDM Methods.

Characteristics	Laaribi	Hanne	Ulengin et al.	Vincke	Felix	Ozernoy	Olson et al.	Ballestero Romero
Problem								
Type of decision problematic	Yes	Yes	Yes	No	No	No	No	No
Problem scale	Yes	No	No	No	No	No	No	No
Potential actions								
Number of alternatives	Yes	Yes	Yes	No	No	No	No	No
New alternative consideration	No	No	No	No	No	No	No	No
Incompatibility and conflict of alternatives	No	Yes	No	No	No	No	No	No
Alternatives' organization	No	No	No	No	Yes	No	No	No
Alternatives' set nature	Yes	Yes	No	No	No	Yes	No	No
Criteria								
Data type	Yes	Yes	Yes	No	No	No	No	No
Measure scale	Yes	Yes	No	No	No	No	No	No
Criteria weighting	Yes	Yes	Yes	No	No	No	Yes	No
Criteria interaction	No	No	No	No	Yes	No	No	No
Usage								
Tool	No	No	No	No	No	No	Yes	No
Approaches for giving partial and final evaluations	No	Yes	Yes	No	Yes	No	Yes	No
Easiness of use	Yes	Yes	No	No	No	No	No	No
Costs for implementing	No	Yes	No	No	No	No	No	No
Decision maker preferences	No	Yes	No	No	No	No	No	No

With regard to *problem scale*, only Laaribi's approach [Laaribi, 2000] describes this parameter. The problem dealt with is estimating spatial impact in a geographic area. It can take two forms: punctual or local (impacts are located) and regional or national (impacts are a large extent).

Next parameter (*new alternative consideration,*) is not considered by the existing approaches. *Incompatibility and conflicts of alternatives* are studied in [Hanne, 1999]. *Alternatives organization* is taken into account in [Felix, 1995]. [Hanne, 1999], [Olson *et al.*, 2000] and [Ulengin *et al.*, 2000] suggest taking into account the nature of alternatives set: continuous or discrete.

Quantitative and qualitative *data types* are studied in [Laaribi, 2000] [Ulengin *et al.*, 2000]. Laaribi [Laaribi, 2000] calls it information nature on criteria that is one of MCDM characteristics. It may be cardinal (quantitative) or ordinal (qualitative). In [Ulengin *et al*., 2000], the authors propose two possible values: quantitative value and qualitative or mixed value (both quantitative and qualitative). In [Hanne, 1999] Hanne considers the discrete, integer or binary, stochastic or fuzzy variables.

Measure scale is considered in approaches [Hanne, 1999] and [Laaribi, 2000]. In [Laaribi, 2000], this is a parameter of the problem situation. For [Hanne, 1999] criteria scale defines a MCDM method validity.

Criteria weighting is taken into account in Laaribi [Laaribi, 2000], Hanne [Hanne, 1999], Ulengin et al. [Ulengin *et al.*, 2000], and Olson et al. [Olson *et al.*, 2000]. Researchers analyse this parameter in order to know whether the MCDM methods take into account relative importance of criteria.

Criteria interaction is considered as relationships between goals in [Felix, 1995]. In fact, in the Felix approach [Felix, 1995], goals serve to choose between the possible decisions that are characterized by their contribution to goals achievement.

The analysis of *tool* is present in [Olson *et al.*, 2000] A tool indicates the possibility to automate the major steps of decision-making.

Approaches for giving partial and final evaluations are compared in [Felix, 1995], [Hanne, 1999], [Olson *et al.*, 2000], and [Ulengin *et al*., 2000]. For Hanne, this parameter describes solution concepts. The approach in [Ulengin *et al.*, 2000] uses these features in order to describe the preference structure and preference method used by a user. Therefore, these authors do not indicate how it is related to the problem situation. Olson et al. analyzes them according to the decision-making steps. Felix analyzes the consistency of evaluations.

Easiness of use is studied by Laaribi and Hanne. Laaribi [Laaribi, 2000] suggests this parameter on the second step of selection and characterises it as the following: easiness of understanding, of programming etc. Therefore, he does not attribute the easiness degree to MCDM methods. Hanne [Hanne, 1999] considers the ease of use as interactivity of man-machine dialogue and "user-friendliness". This approach does not explain how to estimate the easiness of use.

The *costs for implementing* and *decision maker preferences* are considered only in [Hanne, 1999].

2.5.3.3. General Remarks

The state-of-the-art presented above reports on our analysis of the existing selection and/or comparison approaches. The analysis shows clearly that there is not a single approach that matches all situations: each approach has advantages and disadvantages that may change in different contexts. The given state-of-the-art allows us:

- to have an overview of the existing selection approaches and to understand their similarities and differences,
- to elicit comparison criteria for the MC method selection,
- to define what a "good" MC method selection approach should be,
- and, to define open issues.

Different approaches are used for different purposes such as:

1. The following approaches: [Hanne, 1999], [Laaribi, 2000], [Ozernoy, 1996], [Ulengin *et al*., 2000] and [Vincke, 1995] ***for selecting a MC method***. If the interaction between goal is important the Felix approach guides through the selection of an adequate MC method.
2. All approaches are useful for ***better understanding of MC method***s. If a decision maker has a good knowledge of alternative methods, he can use Vincke's approach.
3. A ***tool facilitating the selection of MC methods*** is recommended when the problem is repetitive (Ozernoy and Ulengin et al.'s approaches).

The whole set of analysed characteristics supports decision-making by specifying important factors to consider. These factors could be studied for choosing a method adapted to the situation at hand.

2.6. Conclusion

Some research works to resolve DM problems are reported in the ISE literature. The main characteristic of these works is that they address one specific DM problem each time.

Given that different DM methods can be used more or less efficiently in different situations, the issue of selecting a method adapted to a specific problem situation must be considered carefully. An incorrect choice of a DM method will lead to poor decisions. In the context of Information Systems Engineering, a bad choice of a DM method can generate organizational disorder (e.g. in the context of Enterprise Architecture), poor project arbitration (e.g. in the context of project portfolio management), poor design choices (e.g. in the context of requirements prioritization), with financial, social, and technical consequences for the enterprise and its IS.

The DM methods application is still very limited due to two main difficulties. Firstly, DM methods coming from the operational research are not expressed in terms used in the IS field. Secondly, the IS engineers are confronted with some practical difficulties. No work has been done so far to understand how, when, or which of these methods could be used during ISE activities. Besides, the

reviewed literature highlights that there is no formalized processes to guide IS engineers through DM activities in order to provide guidelines for proper DM and also for accompany IS engineers with the explanation of the additional activities as specifying DM situation, identifying possible alternatives, criteria and so on.

Based on this analysis, a "better" approach than the existing ones might be elaborated. Such an approach should be able to deal with the following problems:

1. Take into account the problem situation,
2. Allow a typology of problem characteristics, take into account data diversity (types, scales etc.),
3. Consider all main groups of the existing DM methods and be able to deal with a new one,
4. Allow selecting of DM method, as well as its better understanding and construction adapted to a concrete case,
5. Take into account interaction between goals,
6. Be structured,
7. Be universal with respect to different application domains,
8. Permit a capitalization of selection results,
9. Suggest a tool facilitating DM methods selection/construction.

Chapter 3: Overview of the MADISE Approach

3.1. Introduction

The goal of this thesis is to provide methodological guidelines for making decisions in Information System Engineering. Indeed, several decisions are costly and risky and may have important consequences. In these cases, it is important that decisions are made in a structured manner in order to enhance their quality and transparency.

As detailed in the Introduction section, the main research question is stated as follows:

> *How methodological knowledge about decision-making can be represented to facilitate the inclusion of decision-making processes into ISE methods?*

The goal is to offer an approach for enhancing decision-making in Information systems engineering based on the method engineering principles and taking into account the specific project situation. This approach is composed of two main elements:

- MADISE DM Ontology
- MADISE DM Method Family

The goal of the DM Ontology (DMO) is to provide an exhaustive modeling of DM concepts and their relationships. As we noted it in the previous chapter, these concepts come from a different scientific field and therefore, their usage in ISE is still limited. In Part II, we present a DM ontology that aims at formalizing DM knowledge. According to the needs of our research, the DM ontology is used for formalizing DM situations, specifying DM requirements, and specifying DM method components. Thus, it offers various elements allowing representing DM fundamentals in terms accessible for ISE, such as UML diagrams. However, its application is wider as it can be used in other fields such as operational research, organizational DM, and so on.

The second element is the DM method family. Its aim is to carry out the reengineering of DM methods in order to make them usable within ISE. For this purpose, we apply SME foundations to DM methods using the DM ontology. Main DM methods and the associated activities (for instance, alternatives and criteria identification) are decomposed in method components, which are organized into a DM method family. The family is completed by two processes used for the contextualization and configuration of a particular DM method from this family depending of the situation at hand and of the ISE methodology.

Figure I.3.1 presents an overview of the MADISE approach.

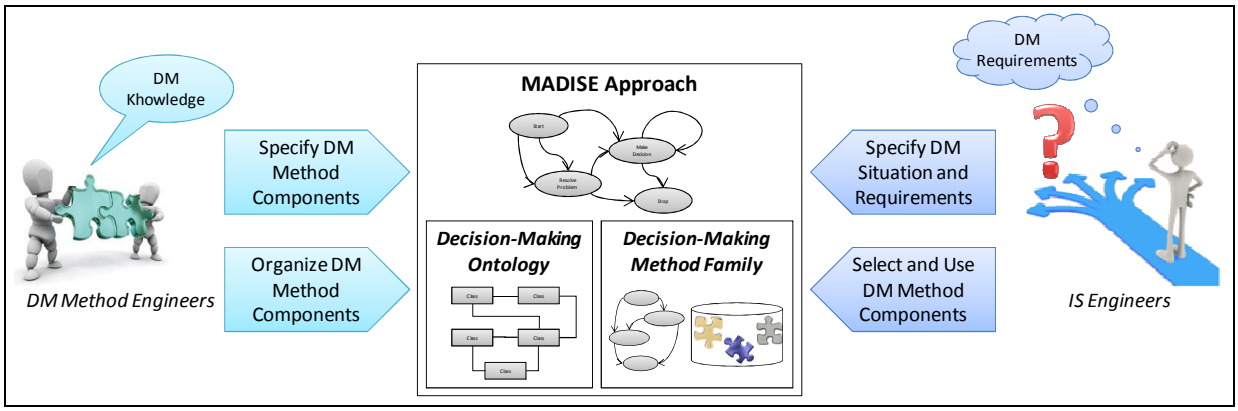


Figure I.3.1. Overview of the MADISE Approach.

As the figure shows, the MADISE approach is composed of two main parts: the DM ontology and the DM method family. A process is offered for configuring the DM method family based on the concepts defined in DMO.

Two kinds of users could use this approach. The first one is composed of DM method engineers who specify DM method components and organize them within the method family. DM engineers, possessing DM knowledge, transform it into DM method components. They use the DM ontology in order to represent DM knowledge and to specify DM methods as components. DM components are stored in a methodological library that we call the MADISE repository. These components are organized within the MADISE repository according to the intention-oriented MADISE process.

The second group represents IS engineers who need DM assistance. They specify their requirements for decision-making and select one or more DM method components in order to enhance their methodologies or to create a new DM application method. An IS engineer has one of the two following goals: (i) to construct a customized DM method or (ii) to extend an IS engineering method with DM components. In both cases, the process starts by specifying requirements for DM, namely the situation and the intention. For this purpose, the DM ontology is used. Then, the engineer selects one or more DM components adapted to the given situation. If DM components are numerous, they are assembled into a composite one. In this manner, the IS engineer obtains a customized DM method fitting the given DM problem (the first goal is reached). Then, if he wants to extend the IS engineering method (the second goal), he integrates the selected or composed component into the ISE method using one of the available SME approaches. Thus, two main cases of the MADISE

approach could be identified: improving an existing ISE method or creating a customized DM method (Figure I.3.2.).

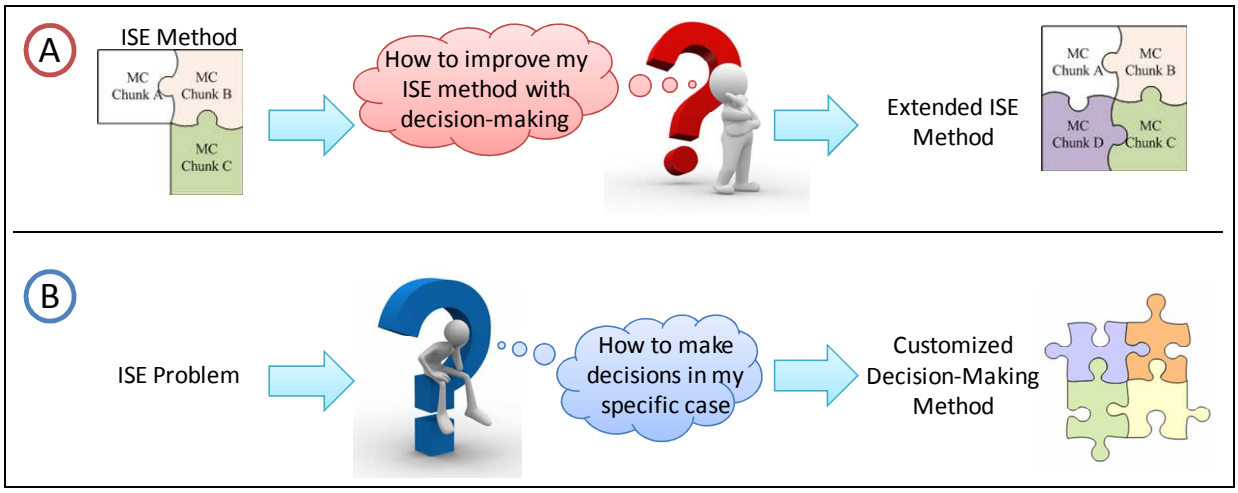


Figure I.3.2. Using the MADISE Approach

The main contributions of this thesis are the MADISE DM Ontology (Part II) and the MADISE DM Method Family (Part III) summed up in the following sections.

3.2. MADISE DM Ontology

The practical needs for DM as well as the number of researches dealing with decision-making in ISE increasingly grow [Kou *et al.*, 2009]. As DM becomes widespread in the ISE field, it is necessary to build a representation of DM concepts and their relations with DM problems in ISE that is shared by researchers and practitioners.

DM knowledge is often expressed through basic definitions of concepts such as alternatives, criteria, decision matrix, and decision itself. However, this domain is much richer, and many other related notions are useful to make right decisions: preferences, weights, thresholds, and so on. This knowledge must be formalized within a model.

We have selected the ontology representation as the most appropriated way of modeling DM knowledge. We have analysed different aspects useful for building ontologies: ontology definition, usage, possible ways of modeling ontologies, ontologies' classifications, goals and elements [Gruber, 1993] [Akkermans *et al.*, 2006] [Batanov *et al.*, 2007] [Kaiya *et al.*, 2006] [Sánchez *et al.*, 2007] [Wieringa *et al.*, 2006].

We have used the ontology fundamentals and DM knowledge (detailed in the State-of-the-Art Chapter) in order to develop the *DM ontology* (DMO) which is a domain knowledge lightweight ontology including concepts, attributes and relationships. This ontology represents DM knowledge as a UML class diagram. We have validated DMO by applying it to the well-known DM method REDEPEND [Maiden *et al.*, 2002] from the requirements engineering field.

3.3. MADISE DM Method Family

We introduce in this thesis the concept of ***method family***. We put forward the method family as a new representation of methods. Current approaches of method engineering are based on the assumption that a method has to be composed "on the fly" during a specific project. Vice versa, we believe that it is possible to anticipate most of the variations of methods and to explicitly represent them through the notion of method family. The method family aims to gather method components from the same field. The construction of a specific method to be used in a specific project is obtained by configuring the family, i.e. by selecting the variants fitting the situation at hand.

We have elaborated the concept of method family (detailed in Chapter 2 of Part III) based on the following existing elements:

- *COMET Meta-Model* as a mean of modular modeling of methods [Kraiem *et al.*, 2008] based on the notion of *Method Chunk* (the representation of method components from the assembly-based SME approach) [Rolland *et al.*, 1998] [Ralyté *et al.*, 2003],
- *Map Model* [Rolland *et al.*, 2001] for the representation of method families and organization of method components within the method family.

In this thesis, we developed a DM Method Family that is composed of DM method components supporting the various activities to be performed to make decisions based on the different available DM approaches. The DM method family includes a collection of DM method components organized into a family in order to allow its further configuration according to a given situation.

The DM method family is defined based on the DM ontology and the method family concept described in Part II:

- The DM method components are specified using the elements of the DM ontology and the component meta-model;
- These components are organized into a family using the method family meta-model and the Map modeling formalism.

In this thesis we show how two known DM approaches, namely the cost-value approach for requirements prioritization [Karlsson *et al.*, 1997], and the tool selection from the Rational Unified Process (RUP) [RUP, 2007] are represented in the DM method family.

In order to support the use of the DM method family, there are two research issues that need to be dealt with:

- How to characterize the situation in such a way that this facilitates the retrieval of the adequate variants in the family;
- and, how to configure the family to generate the specific method to be applied for the given situation

We answer these two questions by including in the MADISE approach a taxonomy of situations and guidelines for performing the contextualization and the configuration of the DM method family.

The DM method family contextualization approach is detailed in Part III (Chapter 3). For this purpose, we have defined a typology of situations appropriated to the field of DM and a process to characterize the context of a DM problem.

The DM Method Family Configuration process is detailed in Part III (Chapter IV). The DM method family configuration provides the means to define a DM application method that is to say to construct a DM method adapted to the given situation. The DM Ontology plays an important role in the configuration process by helping in formalizing the specific DM situation and by identifying requirements for making a decision. The method family configuration process is modeled with the Map formalism.

3.4. Synthesis of the Thesis Results

The results of this thesis are summed up at Figure I.3.3.

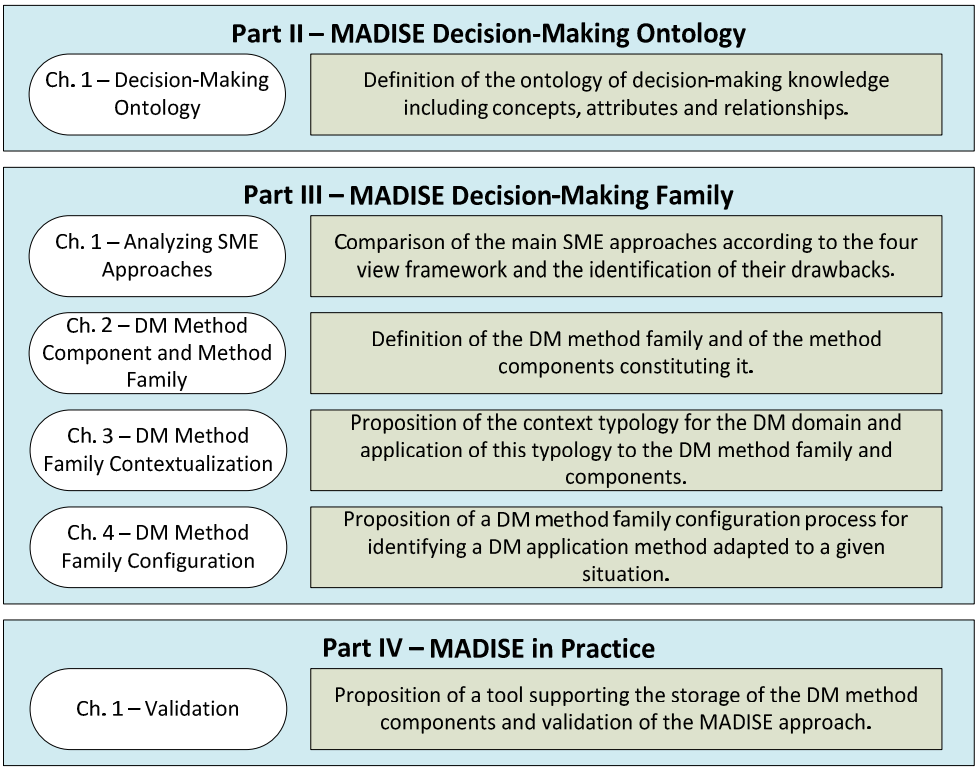


Figure I.3.3. Synthesis of Thesis Results.

Part II. Decision-Making Ontology

Before proceeding to analyze different DM methods, it is necessarily to provide a common conceptual basis which could be used for different purposes, such as identifying needs for DM, characterizing situations requiring DM, formalizing DM methods. For these reasons, we have studied different DM concepts and organized them within an ontology presented in the following Chapter.

Chapter 1: Decision-Making Ontology

1.1. Introduction

The *DM ontology* (DMO) is a representation of DM concepts for formalizing DM knowledge. The goal of this Chapter is to describe the DM ontology. Even if the number of researches dealing with DM in IS engineering increasingly grows [Kou *et al.*, 2009], a complete DM ontology does not exist. As DM becomes widespread, it is mandatory to build a shared representation of DM concepts and to show how these concepts are related to DM problems in IS engineering.

The main goal of DMO is to represent concepts of the DM domain, as well as their properties and relations for their further use within IS engineering. In our approach, the main usage of the DM ontology is twofold: (i) to clarify DM concepts for formalizing DM situations and specifying DM requirements and (ii) to specify DM method components.

This Chapter contains the description of the DM ontology: the application of the ontology fundamentals to DMO, the general view of DMO, the description of its elements, and an example for illustrating DMO.

DM knowledge is often represented as a set of basic definitions such as alternatives, criteria, decision matrix, and decision itself. However, this domain is much richer, and many other related notions are useful for make right decisions: preferences, weights, thresholds, and so on. This knowledge must be formalized within a model.

There are two potential ways to model DM knowledge: as an object model or as an ontology. In addition, ontologies are often represented as class models (that is to say as object models). Therefore, a difference between ontologies and object models must be specified.

[Batanov *et al.*, 2007] compares two representations according to their elements, modeling languages and implementations. This work indicates close similarity between concepts and state the difference between them at the level of deliverables: "while ontologies represent well structured descriptions (explanation) of mutually related terms, the object model is to represent a system as structure of modules (objects) ready for implementation in software". [Kaiya *et al.*, 2006] describe domain ontology and meta-model in the requirements engineering field and promotes the idea that a domain ontology provides semantic basis on requirements to be elicited in order to obtain the concepts representation shared by all stakeholders. Ontologies could be linked to object models as the former express a formal, explicit specification of a shared conceptualization and the latter ones refer to the collection of concepts used to describe the generic characteristics of objects in object-oriented languages [Batanov *et al.*, 2007]. In that way, we deduce that the main distinction relates to the semantic nature and the shared conceptual representation of ontologies.

Based on this, we have selected the ontology representation as the most appropriated way of modeling DM knowledge. The following chapter shows how the ontologies' fundamentals are applied to the DM domain. We finish this Chapter by applying the DM ontology to two cases: the ERP tools selection and the REDEPEND approach of the requirements prioritization.

1.2. Ontology Fundamentals Application to DMO

The DM ontology is developed based on DM knowledge and the State-of-the-Art on ontologies fundamentals. This section covers the usage of ontology for representing DM knowledge and emphasizes the main elements: ontology definition, usage, goals, ontologies' classifications, elements, and possible ways of modeling ontologies. These elements are necessary for constructing the ontology for decision-making.

1.2.1. Ontology Definition

The term ontology is taken from philosophy, where *Ontology* is a systematic account of *Existence*. The notion of "ontology" denotes the science of being and, with this, of descriptions for the organization, designation and categorization of existence [Rebstock *et al.*, 2008]. Gruber was the first to formulate the term ontology in the field of Computer Science [Gruber, 1993] and defined it as "an explicit specification of a conceptualization".

[Gruber, 1993] has found the main principles for constructing ontologies in Computer Science and defined the main ontology elements such as classes, relations, functions, or other objects. Since then, many approaches were developed for creating and applying ontologies [Akkermans *et al.,* 2006] [Batanov *et al.,* 2007] [Hevner *et al.,* 2004] [Kaiya *et al.,* 2006] [Sánchez *et al.,* 2007] [Wieringa *et al.,* 2006]. For instance, [Rebstock *et al.,* 2008] defines the ontology applied to Computer Science as "the capture of the recognized and conceived in a knowledge domain for the purpose of their representation and communication". For [Sánchez *et al.,* 2007], an ontology is a way of representing a common understanding of a domain. [Akkermans *et al.,* 2006] considers ontology as a novel and

distinct method for scientific theory formation and validation. However, all new definitions are based on the idea that Computer Science ontology is a way of representing concepts like in the Gruber's approach [Sánchez *et al.,* 2007].

1.2.2. Ontology usage

[Sánchez *et al.,* 2007] gives three examples of Computer Science domains that use ontologies: Artificial Intelligence, Software Engineering and Database systems. Another field is Web ontologies [Martín, 2003]. Applied to these fields, two main usages of ontologies can be defined [Akkermans *et al.,* 2006] [Sánchez *et al.,* 2007]:

- Ontology as a theoretical model of a real-world domain as it has an explicit content or substantive reference;
- Ontology as a computational specification for computer information systems.

These two usages allow better comprehension of the ontology term.

The first one deals with the ontology representation as expression of the structure and characteristics of things [Akkermans *et al.,* 2006] [Sánchez *et al.,* 2007]. In Computer Science, the mechanism used to show the structure has always been the creation of models [Sánchez *et al.,* 2007]. The graphical and diagrammatic representations may be employed for ontology representation. [Kaiya *et al.,* 2006] views domain ontologies including a thesaurus in a given problem domain.

The second one allows conceptualization of real-world knowledge [Akkermans *et al.,* 2006]. A conceptualization is an abstract, simplified view of the world that we wish to represent for a purpose [Gruber, 1993]. Conceptualization is the process by which the human mind forms its idea about a part of the reality. This idea is a mental representation free of accidental properties and based on essential characteristics of the elements [Sánchez *et al.,* 2007]. Conceptualization allows identifying main elements (objects, their types, properties and relationships) in order to formalize the body of knowledge in the given domain by discarding elements that are not important for a given purpose. For [Akkermans *et al.,* 2006], this second use is the real values of ontologies and the authors go further as they claim ontologies must provide actionable knowledge taking into account domain learning and involving system dynamics.

1.2.3. Defining the DMO Goals

In general, the abstract representation of phenomena expressed with a model must be relevant to the model's purpose [Offen, 2002]. Ontology is a conceptualization, which is an abstract, simplified view of the world that we wish to represent for some purpose [Gruber, 1993]. We have identified the following goals based on the reviewed literature on ontologies in Computer Science [Gruber, 1993] [Akkermans *et al.*, 2006] [Kaiya *et al.*, 2006] [Sánchez *et al.*, 2007] [Offen, 2002] [Gómez-Pérez, 2001]:

- **Knowledge conceptualization.** As ontologies represent concepts, they offer "ways to model phenomena of interest, and, in particular, model theories that are cast in the form of a conceptual framework in a much more rigorous fashion" [Akkermans *et al.*, 2006].
- **Domain modeling.** Ontology aims at modeling a specific domain [Sánchez *et al.*, 2007]. [Offen, 2002] claims that "domain understanding is the key to successful system development". Domain understanding is usually represented via some form of domain modeling [Offen, 2002]. This allows representing complex real world objects with graphical and diagrammatic representations understandable and accessible to experts and practitioners in diverse domains.
- **Anchoring** [Offen, 2002]. Ontology allows anchoring concepts – often abstract – to concrete application domains.
- **Sharing representation.** Agents must communicate about a given domain and have a common language within this domain. [Gruber, 1993] calls this "ontological commitments". An ontology must include atomic concepts that all stakeholders, or other agents can commonly have in a problem domain [Kaiya *et al.*, 2006]. From this view-point, ontology is a compromise between different viewpoints, different stakeholders, or involved parties. [Sánchez *et al.*, 2007] claims that "not only people, but also applications must share a common vocabulary, that is, a consensus about the meaning of things". This consensus is reached by building ontologies, which are one of the solutions for representing this common understanding.
- **Model validation.** In Software Engineering, a specific ontology could be taken as reference point to validate a model that acts over a particular domain [Sánchez *et al.*, 2007]. The following criteria may be tested: consistency, completeness, conciseness, expandability and sensitiveness [Gómez-Pérez, 2001].

Based on this, goals for building DMO must be specified. We have applied the goals, identified from the reviewed literature on ontologies in Computer Science [Gruber, 1993] [Akkermans *et al.*, 2006] [Kaiya *et al.*, 2006] [Sánchez *et al.*, 2007] [Offen, 2002] [Gómez-Pérez, 2001] to the DM field. The goals of DMO in our approach are the following:

- **Knowledge conceptualization.** The DM field contains many terms related to each other by complex relationships. To understand them, it is necessary to create an abstract, simplified view of the DM knowledge that is a conceptualization. Creating an ontology allows providing a flexible way for conceptualizing DM knowledge.
- **Domain modeling.** In the literature, DM models are often limited to some basic elements such as alternatives, criteria, weights, DM actors, and sometimes DM problem [Roy, 1985]. However, this field includes other concepts: preferences, thresholds, decision's future consequences and so on. In this case, the motivation for DMO is to have a unique model for DM knowledge.
- **Anchoring.** DM concepts come from the operational research field. The situation is frequent when it is difficult to say which DM concept corresponds to the given ISE concept. In this case, DMO aims at relating DM concepts to IS engineering.

- **Sharing representation.** All participants of the DM process (such as IS engineers, method engineers, users, and stakeholders) and also applications (as system actors) must share a common understanding of the DM problem.
- **Model validating.** The DMO application enables validating existing DM models or new ones in the ISE domain. For instance, the DM models from the ISE domain could be evaluated according to their completeness, consistency and so on.

1.2.4. Classifying DMO with Ontology Classifications

[Sánchez *et al.*, 2007] presents several classifications of ontologies in Computer Science:

1. by their level of generality (by Guarino N., 1998):

- top-level ontologies, which describe domain-independent concepts such as space, time, etc., and which are independent of specific problems;
- domain and task ontologies which describe, respectively, the vocabulary related to a generic domain and a generic task;
- and, finally, application ontologies, which describe concepts depending on a particular domain and task.

2. by their use (by Van Heijst G., Schereiber A.T. and Wieringa B.J., 1996):

- terminological ontologies, which specify which terms are used to represent the knowledge;
- information ontologies, which specify storage structure data; and
- knowledge modeling ontologies, which specify the conceptualization of the knowledge.

3. Fensel D.'s (2004) classification:

- domain ontologies, which capture the knowledge valid for a particular domain;
- metadata ontologies, which provide a vocabulary for describing the content of on-line information sources;
- generic or common sense ontologies, which capture general knowledge about the world providing basic notions and concepts for things like time, space, state, event, etc;
- representational ontologies, that define the basic concepts for the representation of knowledge; and
- finally, method and particular tasks ontologies, which provide terms specific for particular tasks and methods. They provide a reasoning point of view on domain knowledge.

4. by the level of specification of relationships among the terms gathered on the ontology (by Gómez-Perez A., Fernández-López M. and Corcho O., 2003):

- Lightweight ontologies, which include concepts, concept taxonomies, relationships between concepts and properties that describe concepts.

- Heavyweight ontologies that add axioms and constraints to lightweight ontologies. Those axioms and constraints clarify the intended meaning of the terms involved into the ontology.

According to these works, DMO can be defined as follows:

- **according to the generality level of ontologies** (Guarino N., 1998 and Fensel D., 2004 classifications): DMO is a domain ontology which captures the knowledge valid for ISE domain and describes the vocabulary related to the domain of DM in ISE;
- **according to the ontology use** (Van Heijst G., Schereiber A.T. and Wieringa B.J., 1996 classification): DMO is a knowledge modeling ontology as it specifies the conceptualization of the DM knowledge;
- **according to the level of specification of relationships among the terms gathered on the ontology** (Gómez-Perez A., Fernández-López M. and Corcho O., 2003 classification): DMO is a lightweight ontology which includes concepts, concept taxonomies, relationships between concepts and properties describing concepts. This level of details is sufficient for describing all necessary elements for representing DM knowledge. For this reason, DMO omits axioms and constraints.

1.2.5. Defining the DMO Content

Based on the literature review, the following main ontology elements can be identified in general [Batanov *et al.*, 2007] [Corcho *et al.*, 2005] [Gruber, 1993]:

- **Concept (class).** Classes are the representation of relevant concepts (no matter if they are abstract or specific concepts) in the domain [Gruber, 1993].
- **Attribute (slot, property).** Attributes are characteristics of concept describing its main particularities.
- **Facet.** Facets are attached to classes or slots and contain meta-information, such as comments, constraints and default values [Batanov *et al.*, 2007].
- **Instance.** Instances are used to represent elements or individuals in the ontology [Gruber, 1993].
- **Relation.** Relations represent a type of interaction between concepts of the domain [Corcho *et al.*, 2005]. They include taxonomies [Batanov *et al.*, 2007], functions [Gruber, 1993] and simple associations.
- **Axiom.** Axioms are used to model sentences that are always true [Batanov *et al.*, 2007]. These axioms are used to generate new knowledge and to verify consistency [Gruber, 1993].

DMO is a lightweight ontology including the following main elements: concepts, relations and properties:

- **Concepts** represent objects from real world and reflect the representational vocabulary from domain knowledge [Gruber, 1993]. For instance, one of the most important concepts in DM is *Alternative*.

- **Attributes** (or properties) are characteristics of a concept describing its main particularities, which are concise and relevant to the ontology's goals. For example, the Alternative concept is described by two properties: *type* and *validity*.
- **Relations** are relationships between concepts representing a type of interaction between concepts. Three types of relationships may be used in ontology: association, generalization, and aggregation [Batanov *et al.*, 2007]. An *association* is a bi-directional link between two concepts. A *generalization* is a particular kind of an association factorizing common (to some concepts) elements into a generic concept. In this case, the elementary concepts inherit attributes and relations of the generic concept and have their own properties. An *aggregation* expresses the relation "is composed of". Concepts represent the whole or a part of the whole. In the Alternative example, the Alternative concept is related to four other concepts: to Alternative Set by the *aggregation* relationship, and to DM Object, Criterion, and Consequence by the simple *associations*. An intuitive decision is a particular type of decisions, thus the generalization relationship permits to relate these.

1.2.6. Modeling DMO

[Sánchez *et al.*, 2007] mentions several methods for modeling ontologies such as frames and first order logic, Description Logics, Entity-Relationship (ER) diagrams or Unified Modeling Language (UML) class diagrams. UML is sufficient to model lightweight ontologies [Sánchez *et al.*, 2007] and is a well-known modeling language. For this purpose, the UML notation is already used, for instance, in an ontology for software metrics and indicators for cataloging web systems [Martín, 2003] or for representing domain ontologies and meta model ontology in requirements engineering [Kaiya *et al.*, 2006].

Like in [Martín, 2003] and in [Kaiya *et al.*, 2006], we use the UML class diagram for modeling DMO. Each *class* represents a concept. Concept taxonomies are represented by *generalization relationships*. The concept structure is expressed through the *composition relationship*. Relations between concepts are represented by *association relationships*. Concept properties are *attributes* of the corresponding classes.

1.3. DMO Elements

In this thesis, we foresee modeling DM knowledge as an ontology. This kind of modeling is justified by the necessity to show different semantic links existing between DM concepts. Based on the previous section, we see DMO as a domain knowledge lightweight ontology including concepts, attributes and relationships. This ontology represents DM knowledge as a UML class diagram (See Figure II.1.1.).

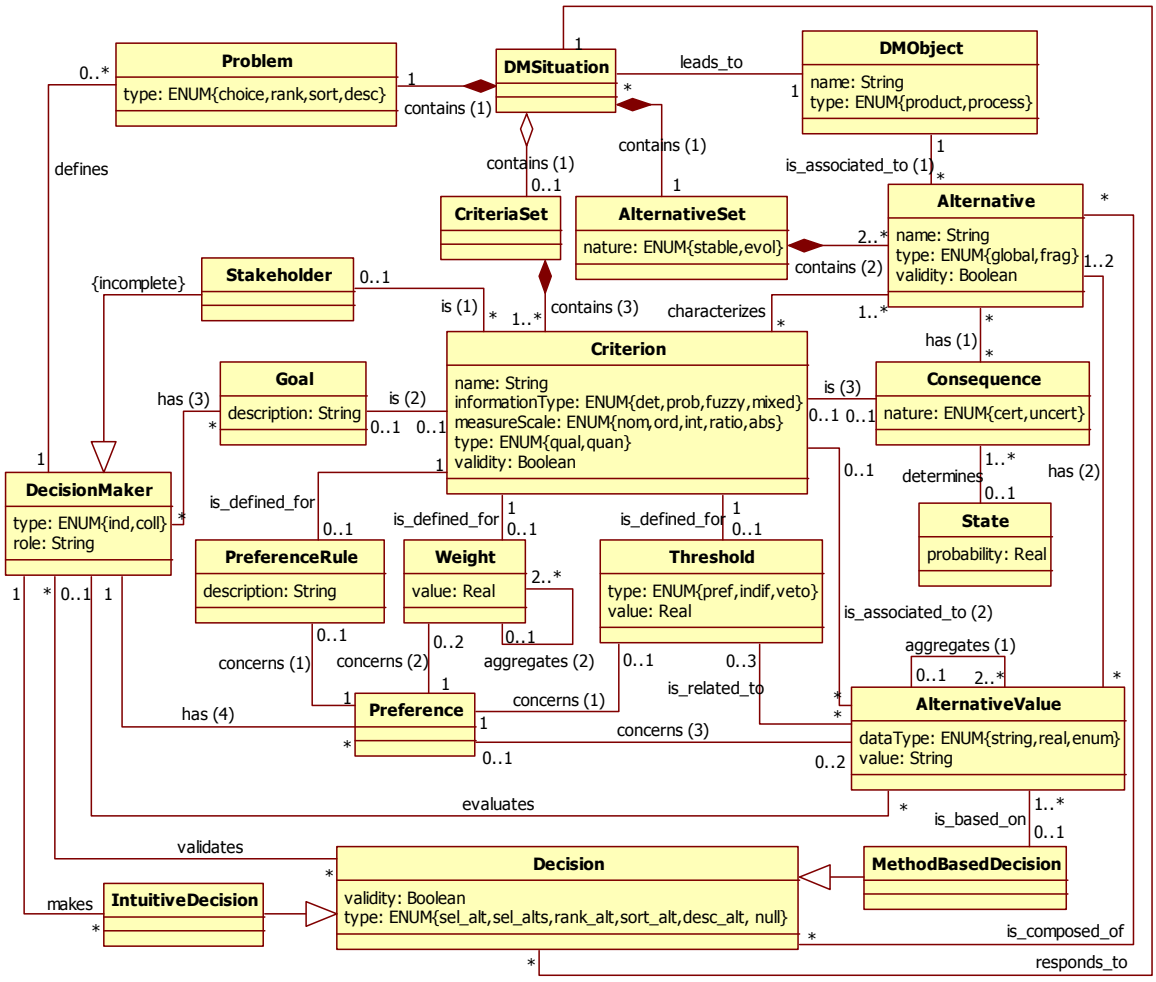


Figure II.1.1. Decision-Making Ontology.

In the following sub-sections, we detail concepts of the DM ontology and we illustrate them using a common example, which is an ERP (Enterprise Resource Planning) project.

1.3.1. Decision

A ***decision*** (See Figure II.1.2.) is an act of intellectual effort initiated for satisfying a purpose and allowing a judgment about the potential actions' set in order to prescribe a final action. A decision (the *Decision* class) is a result of different actions allowing resolving a DM problem formalized as a DM situation (the *DMSituation* class). In this manner, each decision is related to a DM situation by the *responds_to* association. Some different decisions could be made in the given DM situation if different DM methods are used. A decision is characterized by its *validity* and *type*. In fact, a validation of the obtained results could be required at the end of the DM process. Decision makers validate decisions (the *validates* relationship). The *decision type* depends on the decision problem defined in a given DM situation and describes the result really obtained by decision-making. The

decision type can be: one selected alternative (sel_alt), multiple selected alternatives (sel_alts), ranked alternatives (rank_alt), sorted alternatives (sort_alt), and described alternatives (desc_alt). It can also be NULL if a decision in not made. The *is_composed_of* association relates the given decision with corresponding alternatives.

Depending on the DM process nature, there are two kinds of decisions: the intuitive and method-based ones, which inherit the attribute and relationships of the *decision* class (see sections II.1.3.2. and II.1.3.3.).

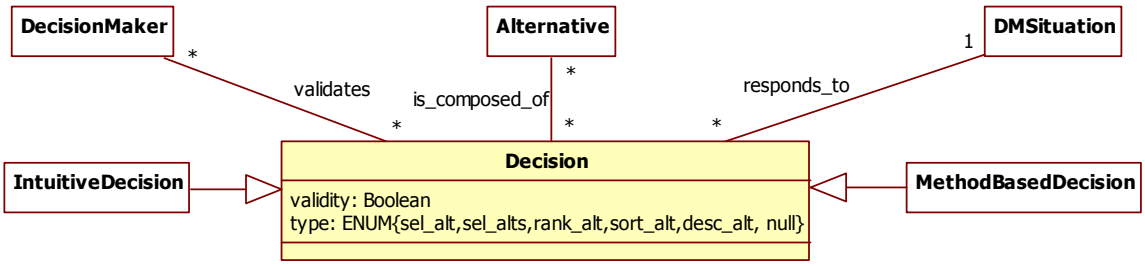


Figure II.1.2. Decision Representation.

Example: Representative examples of decisions are:

- Selected tool (decision type is sel_alt);
- Prioritized requirements (decision type is rank_alt).

1.3.2. Intuitive Decision

Intuitive decisions (the *IntuitiveDecision* class) at Figure II.1.3. represent a particular type of decisions. Actually, many decisions are made on the fly without using any decision-making method. Even if they do not use these methods, decision makers can have some information about the DM situation, for instance, about the alternatives' set. This class is related to the *DecisionMaker* one in order to indicate the decision maker who performs the given intuitive decision (the *makes* relationship).

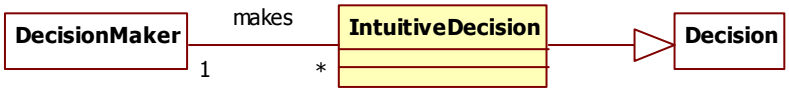


Figure II.1.3. Intuitive Decision Representation.

Example: An engineer must rank ten requirements by their order of implementation. As only one person is available, requirements must be considered one by one. All requirements are

mandatory and none is extremely urgent. In this case, the engineer could rank the requirements according to his intuition.

1.3.3. Method-based Decision

In contrast to the intuitive decisions, the ***method-based*** ones (the *MethodBasedDecision* class) are made using different DM methods (See Figure II.1.4.). For instance, these methods enable defining a criteria set, measuring alternatives according to criteria, aggregating alternative values, and so on. As method-based decisions are based on the defined values of alternatives, the corresponding class is related to the *AlternativeValue* class by the *is_based_on* association.

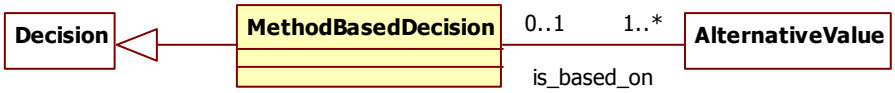


Figure II.1.4. Method-based Decision Representation.

Example: A company has to acquire an ERP tool. The decision must be founded on multiple criteria of various nature, for instance, purchase cost, maintenance cost, confidence in editor, experience with editor, offered functionalities, etc. In order to deal with these criteria and because of the importance of this acquisition, a method is needed in order to provide the best result.

1.3.4. Decision-Making Situation

A ***DM situation*** is a set of specific conditions dealing with a given DM object (See Figure II.1.5.). Hence, a DM situation is firstly characterized by a DM object (the *leads_to* association). It means that the object must be clearly defined in each DM situation. Secondly, a DM situation is a combination of specific conditions of decision-making. These conditions include: a problem, a set of alternatives, and a set of criteria (the *contains (1)* composition link). The problem allows defining the nature of the future decision. Alternatives must be at least two in each DM situation. The problem and the alternatives' set are mandatory: they constitute the minimal requirements for decision-making. The criteria set is not mandatory as a decision can be made without any complementary information about alternatives. Each DM situation can imply several decisions (related to it by the *responds_to* association) if different methods are used for decision-making.

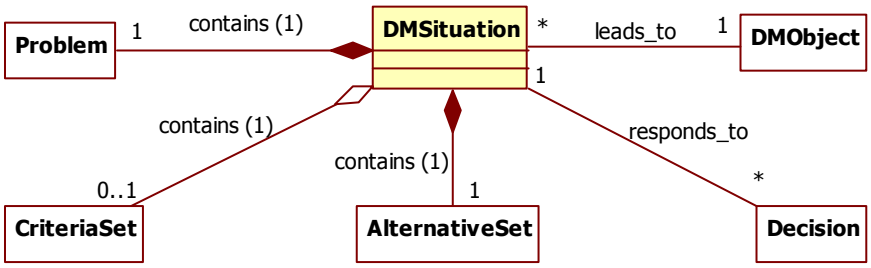


Figure II.1.5. Decision-Making Situation Representation.

Example: Using the previous example of the ERP acquisition, the DM situation is described as follows. The DM object is the ERP tool, the problem is the choice. The potential ERP tools compose the alternative set. The criteria set includes purchase cost, maintenance cost, confidence in editor, experience with editor, offered functionalities. And, finaly, the related decision will be the selected ERP tool.

1.3.5. Decision-Making Object

A ***DM Object*** (the *DMObject* class) represents an artifact being the subject of decision-making in IS engineering (See Figure II.1.6.). This term is important as it refers to the content of the DM process. The same artifact could be the DM object in different DM situations retrieved by the *leads_to* relationship. DM object refers to different alternatives (the *is_associated_to (1)* relationship). The DM object is characterized by a name, describing the object, and a type, which can be a process or a product element in IS engineering.

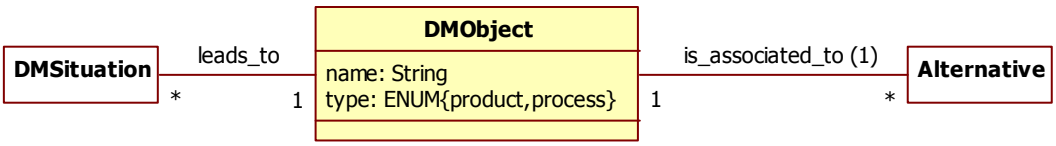


Figure II.1.6. Decision-Making Object Representation.

Example: A usual example of the DM object is *requirement*.

1.3.6. Problem

A decisional ***problem*** (See Figure II.1.7.) characterizes the result expected from DM [Roy, 2005]. The problem is characterized by a problem type that can be: choice, ranking (rank), sorting (sort), and description (desc). When the result consists in a subset of potential alternatives (most often only one alternative), it is a *choice problem.* When the result represents the potential alternative affectation to

some predefined clusters, it is a *sorting problem.* If the result consists in a potential collection of ordered alternatives, then it is a *ranking problem*. When a decision is highlighted only by a description of alternatives and of their impact using an appropriate language, it is a *description problem*. The problem type is usually defined by a decision-maker (the *defines* relationship).

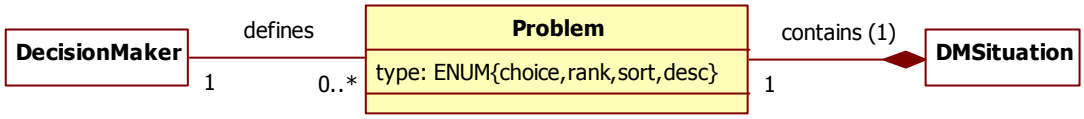


Figure II.1.7. Problem Representation.

Example: In the case of the ERP tool selection, the problem is choice.

1.3.7. Alternative

An ***alternative*** (See Figure II.1.8.) is a possible action available for decision-making (the *Alternative* class). This concept designates the decision object (the *is_associated_to (1)* relationship). Any decision involves at least two alternatives that must be well identified. These alternatives are organized into a set (see the next section). Each alternative can be characterized by a name, a type, and validity. The type shows whether the alternative can be applied with another one (the *fragmented alternative* – frag) or its application is exclusive of other alternatives (the *global* one). The initial alternatives' set can be reviewed in order to select a subset of valid alternatives; that is to say to delete the invalid ones. To differentiate them, the alternative class is described by the *validity* attribute.

Alternatives are characterized by criteria (the *characterizes* association). They could also be described by analyzing future consequences that the alternatives' realization can produce (the *has (1)* relationship). Each alternative is evaluated according to the criteria or to the other alternatives. In order to formalize this information, the *AlternativeValue* class is used. It is related to the *Alternative* class by the *has (2)* association. The *is_composed_of* association is used in order to indicate which alternative(s) is (are) retained for prescribing the final decision.

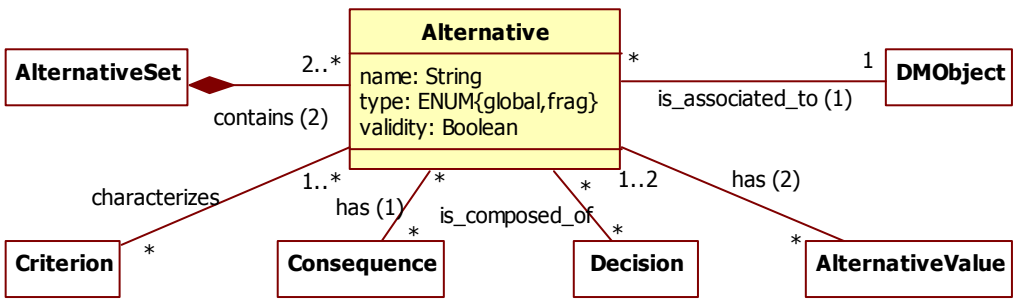


Figure II.1.8. Alternative Representation.

Example: For instance, an example of alternative is the SAP ERP in the case of the ERP tool selection.

1.3.8. Alternative Set

An ***alternative set*** (See Figure II.1.9.) is a set of alternatives available in a given DM situation (the *contains (1)* composition link). This concept allows characterizing the collection of alternatives available in a given situation as a whole. The alternative set is composed of at least two alternatives and is described by its nature (the *contains (2)* composition link). The *nature* means the stable or evolving character of the alternative set. A *stable set* contains the same alternatives during a relatively long period. Alternatives composing an *evolving set* (evol) change often: the existing alternatives are removed and/or the new ones appear.

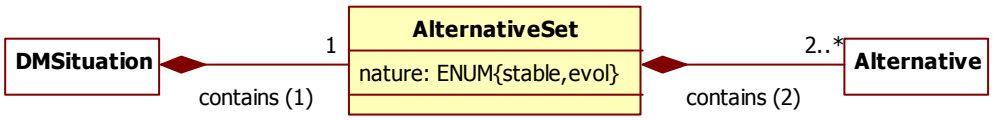


Figure II.1.9. Alternatives' Set Representation.

Example: For the ERP tool selection, the alternative set is composed of SAP ERP, Oracle e-Business Suite, and OpenERP.

1.3.9. Criterion

A ***criterion*** represents information of any kind that enables the evaluation of alternatives and their comparison (See Figure II.1.10.). Criteria organized into a set (the *contains (3)* composition link) compose the DM situation and characterize alternatives (the *characterizes* relationship). Criteria are described by a *name* and the following attributes:

- ***informationType.*** The information type describes the criterion nature: determinist (det), probabilistic (prob), fuzzy, or mixed.
- ***measureScale.*** The measure scale indicates the kind of scale used for measuring alternatives according to the given criterion: nominal (nom), ordinal (ord), interval (int), ratio, and absolute (abs).
- ***dataType.*** The data type is qualitative (qual) or quantitative (quan).
- ***validity.*** As for alternatives, validity shows whether the initially identified criterion is selected or not for evaluating alternatives.

Criteria are related to the *AlternativeValue* class in order to indicate the characteristics that describe the alternatives and to give the corresponding alternative values (the *is_associated_to (2)* relationship). Criteria can have also different origins: alternative characteristics, future consequences of alternatives, goals, and stakeholders.

- ***Alternative characteristics.*** The associated criteria are different characteristics describing alternatives (the *characterizes* relationship). These characteristics can be used in the case when alternatives have intrinsic characteristics, which can be considered as criteria. The engineer analyzes them in order to identify those that are important for arbitrating between alternatives. One example is requirements characterization in [Karlsson *et al.*, 1997].
- ***Future consequences of alternatives.*** In this case, criteria represent possible consequences induced by alternatives in the future. This is a particular kind of alternative characterization as it gives a probable description of the future result that cannot be guaranteed at the moment (the *is (3)* relationship).
- ***Goals.*** Goals that stakeholders have in relation with a given DM problem can also be considered as criteria. For instance, each alternative can be evaluated according to its contribution to goal achievement (the *is (2)* relationship). Goals as criteria are used in [Maiden *et al.*, 2002] for requirements prioritization.
- ***Stakeholders.*** When stakeholders participate in the DM process and have divergent points of views on different elements of the DM situation (alternatives and their values, preferences expressed according to weights, thresholds and preference rules, and so on), they can become criteria as their different opinions are to be taken into account (the *is (1)* relationship).

Criteria can be described by three parameters that are defined by a decision maker: a preference rule (*PreferenceRule*), a weight (*Weight*), and thresholds (*Threshold*) related to each criterion by the *is_defined_for* association. These parameters state different kinds of preferences that decision makers can express towards criteria and are detailed respectively in the sections II.1.3.14, II.1.3.15, and II.1.3.16.

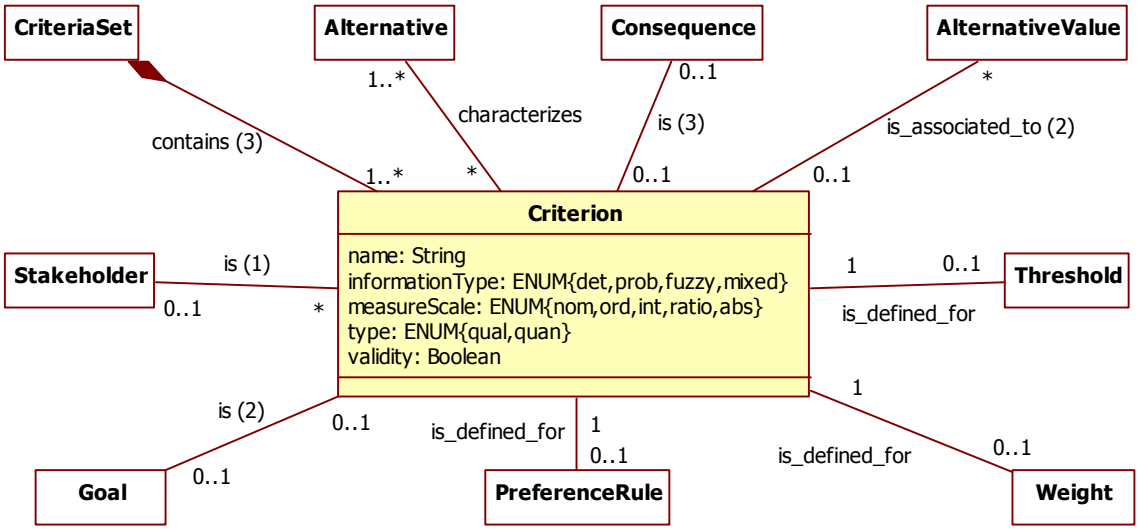


Figure II.1.10. Criterion Representation.

Example: One of the criteria, available for the ERP tool selection, is the purchase cost. This criterion is characterized by the determinist information type, interval measure scale, and quantitative type.

1.3.10. Criteria Set

A ***criteria set*** at Figure II.1.11. (*CriteriaSet* class) presents the set of criteria available in a given DM situation (the *contain (1)* composition link). The criteria set is not mandatory in each DM situation and can contain at least one criterion (the *contain (3)* composition link).

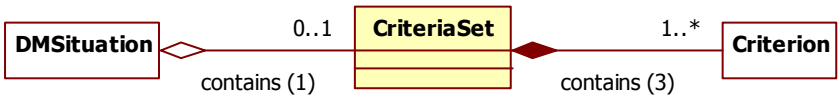


Figure II.1.11. Criteria Set Representation.

Decision-makers can express their preferences without considering any criterion. For this reason, some decision-making models do not contain a criteria set. This is the case, for instance, of the social choice theory [Bouyssou *et al.*, 2009] when decision-makers give their preferences between candidates without comparing them according to any criterion.

Example: In the case of the ERP tool selection, the criteria set includes purchase cost, maintenance cost, confidence in editor, experience with editor, offered functionalities.

1.3.11. State

A ***state*** (the *State* class) describes the nature of the environment affecting alternative consequences in the future (See Figure II.1.12.). It impacts consequences (the *determines* association) and is characterized by the probability in which the state can occur (the *probability* attribute).

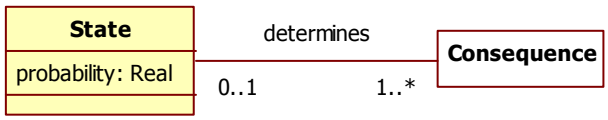


Figure II.1.12. State Representation.

This concept corresponds to the Nature state concept introduced within the decision tree DM model [Kast, 2002].

> ***Example:*** A company, which intends to purchase an ERP tool, plan to open a new branch. In this case, the maintenance cost (including the annual license payment) will increase, as the number of the workstations will grow. The branch opening depends on the company market capitalization the next year. The probability that the market capitalization increases is 70%.

1.3.12. Consequence

A ***consequence*** (the *Consequence* class) expresses the impact that an alternative can have whether it is realized following the made decision (See Figure II.1.13.). The consequence is described by its *nature*: certain (cert) or uncertain (uncert). As stated above, a consequence characterizes alternatives (the *has (1)* association), represents a criterion (the *is (3)* association), and depends on the future state (the *determines* relationship).

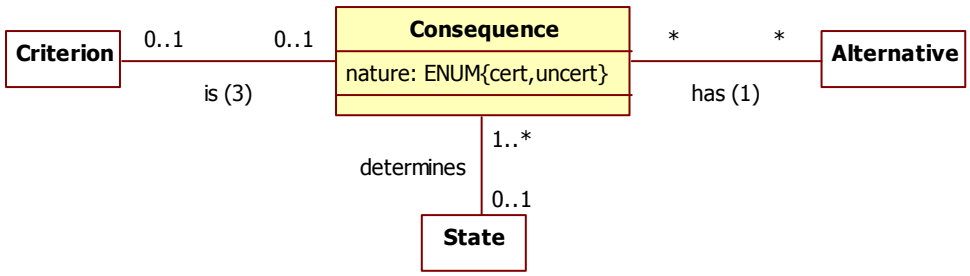


Figure II.1.13. Consequence Representation.

Example: The maintenance cost of the ERP tool depends on several factors (for instance, acquisition of a branch) and implies the consequences of the ERP implementation in the future.

1.3.13. Alternative Value

An ***alternative value*** (the *AlternativeValue* class) (at Figure II.1.14.) represents the evaluation of an alternative (the *has (2)* relationship) according to a criterion (the *is_associated_to (2)* association). It can express an aggregated alternative value (the *aggregates (1)* association) if a value is obtained from other values (considered as elementary) by applying a method. For instance, it can be the sum of values corresponding to different criteria. The alternative value has a *type* (string, real, or enumeration – enum) and a corresponding *value*. An alternative value can be also defined by analyzing the decision-maker preferences. In fact, he can express his preferences between two alternatives by pair wise comparison (the *concerns (3)* relationship to the *Preference* class) or give a value according to a criterion (the *evaluates* relationship to the *DecisionMaker* class). When doing the pair-wise comparisons, the complementary information can be specified as a threshold (the *is_related_to)* association) that shows an acceptable level for preferences between values (see section II.1.3.16. for details on thresholds). Three thresholds (preference, indifference and veto thresholds) could be defined for each alternative value in a given DM situation. The same threshold can be related to many alternatives values. The alternatives values are used as basis for making the method-based decisions (the *is_based_on* relationship).

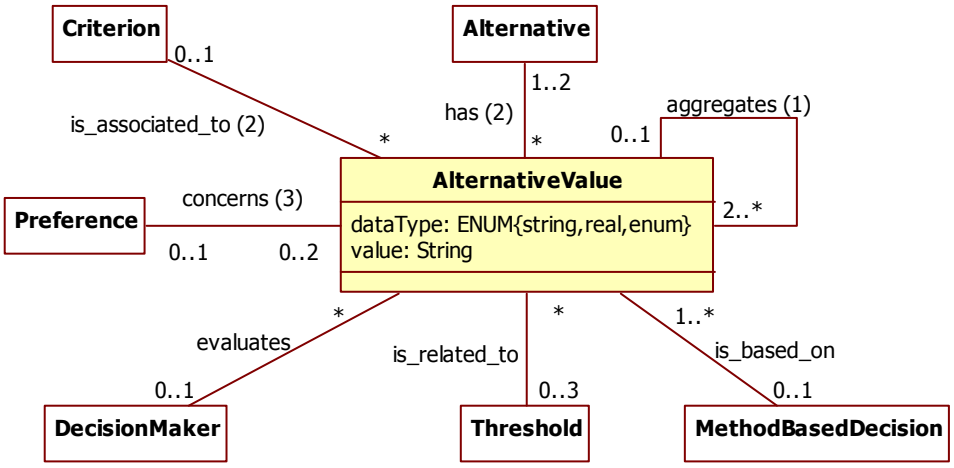


Figure II.1.14. Alternative's Value Representation.

Example: The following table (Table II.1.1.) represents a set of alternative values corresponding to the evaluation of the RE methods dealing with IS security according to the formalism, impact, and expertise.

Table II.1.1. Values of the RE Methods of IS Security.

Criteria	NFR Framework	KAOS	Secure Tropos	GBRAM	Misuse Cases
Formalism	semi-formal	formal	formal	informal	informal
Impact	3	1	3	1	2
Expertise	normal	high	high	normal	low

1.3.14. Preference Rule

A ***preference rule*** (the *PreferenceRule* class) defines a wishful value of a criterion according to a given need (See Figure II.1.15.). It is characterized by its description which shows the decision-maker preference (the *concerns (1)* relationship) for a given criterion (the *is_defined_for* association). It can be, for instance, a function (maximization, minimization), or an ordered list.

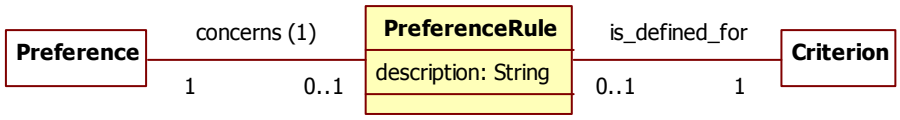


Figure II.1.15. Preference Rule Representation.

Example: For illustrating preference rules, the following examples can be given:

- Maximization function (for the Profit criterion);
- Minimization function (for the Cost criterion);
- Ordered list: High $\succ$ Medium $\succ$ Low, where $\succ$ means "is preferred to" (for the Expertise criterion).

1.3.15. Weight

The ***weight*** concept (the *Weight* class) designates the relative importance of a criterion (See Figure II.1.16.). It is expressed by a numeric *value*. Weights are defined for criteria (the *is_defined_for* relationship) through the analysis of the decision maker preferences. A decision maker preference may concern a weight or two weights in the case of pair-wise comparisons (the *concerns (2)* relationship). A weight value can be obtained by the aggregation of at least two values (the *aggregates (2)* relationship).

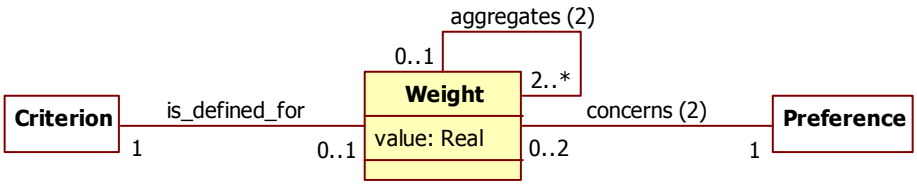


Figure II.1.16. Weight Representation.

Example: For instance, for the ERP tool selection, only three criteria are retained: purchase cost (pc), maintenance cost (mc), and editor reputation (er). The decision-maker defines the following criteria: $W_{pc} = 0,4$; $W_{mc} = 0,5$; $W_{er} = 0,1$.

1.3.16. Threshold

A ***threshold*** (the *Threshold* class) shows an acceptable alternative value (See Figure II.1.17.) expressed for a given criterion (the *is_defined_for* association). There are three types of thresholds: preference (pref), indifference (indif), and veto. A threshold can be related to many alternative values (by the *is_related_to* association). Thresholds are expressed by numeric values and are defined by decision-makers through the *Preference* notion (the *concerns (1)* association).

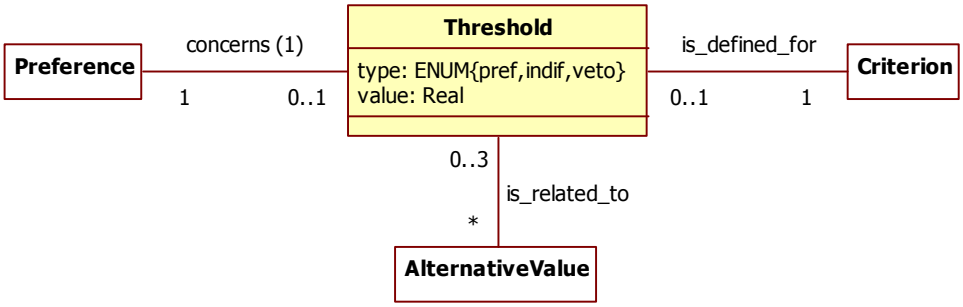


Figure II.1.17. Threshold Representation.

Thresholds are widely used in outranking decision-making methods [Ref].

Example: A veto threshold can be established for the purchase cost of the ERP tool. The tools having the cost greater than the veto threshold, would not be selected for purchasing.

1.3.17. Preference

A ***preference*** (the *Preference* class) at Figure II.1.18. represents an opinion that a decision-maker (the *has (4)* relationship) has on different DM situation elements such as alternatives and criteria. More

exactly, decision-makers can express their preferences on preference rules, weights, thresholds, and, directly, on alternatives (the *concerns (1)*, *concerns (2)*, and *concerns (3)* relationships). Regarding to weights, decision makers can define their preferences for each of them or make pair wise comparisons between two weights. They may also evaluate alternatives one by one, or compare two alternatives. With regards to preference rules and thresholds, they give their preferences only one by one.

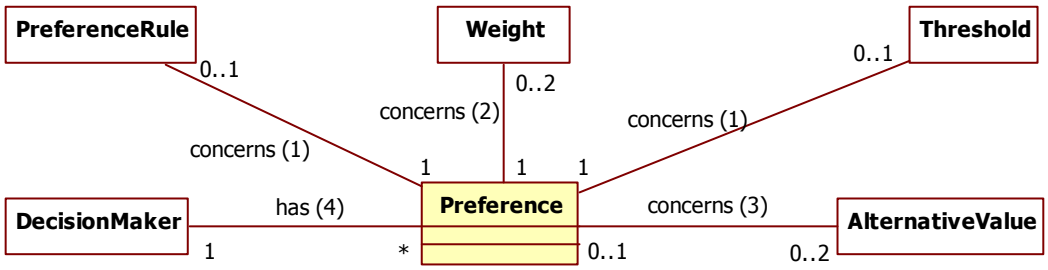


Figure II.1.18. Preference Representation.

For instance, the preference concept is present in the AHP method applied for requirements prioritization [Karlsson *et al.*, 1997] [Maiden *et al.*, 2002].

> ***Example:*** A decision-maker expresses his preferences between several alternatives by giving pair wise comparisons: the requirement X is preferred to the requirement Y.

1.3.18. Decision-Maker

A ***decision maker*** (the *DecisionMaker* class) is a human actor contributing to the DM process at its different stages (See Figure II.1.19.). Each decision maker is characterized by the type that can be individual (ind) or collective (col). An individual decision maker is a human actor that processes one or more DM steps. A collective decision maker is a group of decision-makers having the same goals and preferences and acting as a unique decision-maker. Decision makers can have three main roles (the *role* attribute): stakeholder, IS engineer, and DM staff.

Decision makers can participate in the DM process in the following manners:

- ***Problem definition.*** A decision-maker defines a problem for each DM situation (the *defines* association with the *Problem* class).
- ***Goal definition.*** A decision-maker can have several goals that may be shared by several decision-makers (the *has (3)* association with the *Goal* class). Goals can represent criteria in a DM situation.
- ***Preference definition.*** A decision-maker can have several preferences (the *has (4)* association with the *Preference* class) concerning criteria preference rules, weights, and thresholds, and also alternative values.

- ***Alternative's evaluation.*** A decision-maker can evaluate one or more alternatives (the *evaluates* association with the *AlternativeValue* class).
- ***Intuitive decision-making.*** A Decision-maker can make intuitive decisions (the *makes* association to the *IntuitiveDecision* class).
- ***Decision validation.*** A decision maker validates decisions (the *validates* association to the *Decision* class).

The decision maker as a stakeholder is described in the following section.

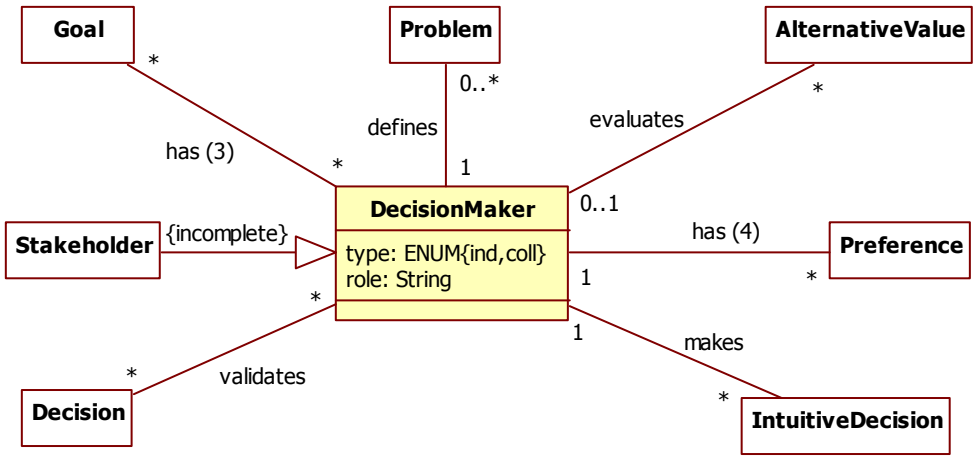


Figure II.1.19. Decision-Maker Representation.

> ***Example:*** An individual decision maker expresses his preferences with relation to the criteria weights in a given DM situation (for instance, requirement prioritization).

1.3.19. Stakeholder

A ***stakeholder*** (the *Stakeholder* class, at Figure II.1.20.) is a particular kind of decision makers who inherits different DM roles: he defines the DM problem, sets goals, expresses preferences on alternatives and criteria and validates the final decision. However, only this type of decision maker could be considered as a criterion in a given DM situation (the *is (1)* relationship). For instance, when several stakeholders give their preferences on different alternatives, they could be considered as criteria and weights could be assigned to them in function of their position in the organizational hierarchy.

Figure II.1.20. Stakeholder Representation.

Example: When several important stakeholders are involved in the DM process, their opinions are not the same. In this case, they may be considered as criteria. Weights could be affected to each stakeholder in order to take into account their importance.

1.3.20. Goal

A ***goal*** (the *Goal* class) illustrated at Figure II.1.21. is an intention or a projected state that a decision-maker intends to achieve. A goal is formalized by a *description*. Goals are expressed by decision makers (the *has (3)* relationship) and can be considered as criteria in a particular DM situation (the *is (2)* relationship). Each alternative can be measured according to its capability to contribute to these goals. In this case, goals become criteria.

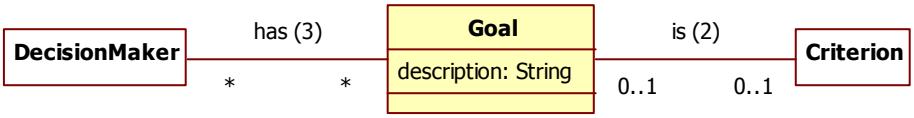


Figure II.1.21. Goal Representation.

Example: An example of the goal is "Increase the market capitalization of the company".

1.3.21. Summary of Decision-making Concepts, Attributes, and Relationships

The concepts, their attributes and relationships are summed up in three tables (respectively in Tables II.1.2., II.1.3., II.1.4.).

Table II.1.2. Decision-Making Ontology: Glossary of Concepts.

Concept Name	***Description***
Alternative	Possible action available for decision-making.
AlternativeSet	Set of alternatives available in a given DM situation.
AlternativeValue	Evaluation of an alternative according to a criterion or an aggregated alternative value.
Consequence	Impact that an alternative can have whether it is realized following the decision made.
CriteriaSet	Set of criteria available in a given DM situation.
Criterion	Information of any kind that enables the evaluation of alternatives and their

	comparison.
Decision	Act of intellectual effort initiated for satisfying a purpose and allowing a judgment about the potential actions set in order to prescribe a final action. It can be an IntuitiveDecision or MethodBasedDecision.
DecisionMaker	Actor contributing to the DM process at its different stages.
DMObject	Artifact being the subject of decision-making in IS engineering.
DMSituation	Set of specific conditions of DM dealing with a given DM object.
Goal	Intention or a projected state that a decision-maker intends to achieve.
IntuitiveDecision	Decision made 'on the fly' without using a DM method.
MethodBasedDecision	Decision based on the application of different DM methods.
Preference	Preference that a decision-maker have on different DM situation elements of alternatives and criteria.
PreferenceRule	Wishful value of a criterion according to a given need.
Problem	Result expected from DM
Stakeholder	Particular role of a decision-maker, which defines the DM problem, sets goals, expresses preferences on alternatives and criteria and validates the final decision.
State	State of the environment affecting alternative consequences in the future.
Threshold	Acceptable value for alternative values expressed for a given criterion.
Weight	Relative importance of a criterion.

Table II.1.3. Decision-Making Ontology: Attributes Description.

Concept	***Attribute***	***Description and/or Domain***
Alternative	name	Name of an alternative.
	type	Type of alternative: global or fragmented.
	validity	Validity of an alternative, which is a Boolean value.
AlternativeSet	nature	Nature of the alternative set, which is stable or evolving.
AlternativeValue	dataType	Type of an alternative value: string, real, or enumeration.
	value	Value of an alternative.
Consequence	nature	Nature of a consequence, which can be certain or uncertain.
Criterion	name	Name of a criterion
	information-Type	Type of information on a criterion: determinist, probabilistic, fuzzy, or mixed.
	measureScale	Measure scale: nominal, ordinal, interval, ratio, and absolute.
	dataType	Type of data: qualitative or quantitative.
	validity	Validity of a criterion, which is a Boolean value.
Decision	validity	Validity of a decision, which is a Boolean value.
	type	The decision type describes the result really obtained by DM. The decision type can be: one selected alternative, multiple selected alternatives, ranked alternatives, classified alternatives, described alternatives, and NULL.
DecisionMaker	type	Type of a decision-maker, which can be individual or collective (a group of decision-makers having the same goals and preferences and acting as a unique decision-maker).
	role	Decision makers can have three main roles: stakeholder, IS engineer, and DM staff.
DMObject	name	Name of a DM object.
	type	DM object type which can be a process or a product element in IS engineering.
Goal	description	Description of a goal.
PreferenceRule	description	Description of preference, for instance, a function (max, min), or an ordered list.

Problem	type	Problem type, which can be choice, ranking, sorting, description.
State	probability	Probability of a state realization in the future.
Threshold	type	Threshold type: preference, indifference, or veto thresholds.
	value	Numeric value of a threshold.
Weight	value	Numeric value of a weight.

Table II.1.4. Decision-Making Ontology: Relationships Description.

Relationship Name	***Source Class***	***Target Class***	***Description***
aggregates (1)	AlternativeValue	AlternativeValue	An alternative value can be an aggregation of at least two values which describe this alternative according to different criteria.
aggregates (2)	Weight	Weight	A weight value can be an aggregation of at least two values.
characterizes	Criterion	Alternative	Each criterion characterizes one or more alternatives. Each alternative can be characterized by one or more criteria.
concerns (1)	Preference	Threshold, PreferenceRule	A preference may concern a threshold or a preference rule.
concerns (2)	Preference	Weight	A preference may concern a weight or two weights in the case of pair-wise comparisons.
concerns (3)	Preference	AlternativeValue	A preference may concern an alternative value or two alternative values in the case of pair-wise comparisons. An alternative value may be defined by a preference of a decision-maker.
contains (1)	DMSituation	Problem, AlternativeSet, CriteriaSet	Each DM situation contains a problem and an alternative set and can contain a criteria set.
contains (2)	AlternativeSet	Alternative	An alternative set contains a least two alternatives.
contains (3)	CriteriaSet	Criterion	A criteria set contains one or more criteria.
defines	DecisionMaker	Problem	A decision-maker defines a problem for each DM situation. He can define several problems for different DM situations.
determines	State	Consequence	A state can determine one or more consequences.
evaluates	DecisionMaker	AlternativeValue	A decision-maker can evaluate one alternative or give a pair-wise evaluation of two alternatives.
has (1)	Alternative	Consequence	An alternative can have one or more consequences; consequences are related to alternatives.
has (2)	Alternative	AlternativeValue	An alternative can have several values; each alternative value is related to one or two alternatives.
has (3)	DecisionMaker	Goal	A decision-maker can have several goals. The same goal may be shared by several decision-makers.
has (4)	DecisionMaker	Preference	A decision-maker can have several preferences.

is (1)	Stakeholder	Criterion	A stakeholder can be a criterion in one or more DM situations.
is (2)	Goal	Criterion	A goal can be a criterion in a given DM situation.
is (3)	Consequence	Criterion	A consequence can be a criterion in a given DM situation.
is_associated_to (1)	Alternative	DMObject	None or several alternatives are associated to a DM object.
is_associated_to (2)	AlternativeValue	Criterion	None or several alternative values can be associated to a criterion.
is_based_on	MethodBasedDecision	AlternativeValue	A method-based decision is based on one or more alternative values.
is_composed_of	Decision	Alternative	A decision can be composed of none, one or more alternatives.
is_defined_for	PreferenceRule, Threshold, Weight	Criterion	A preference rule, a threshold, or a weight can be defined for a criterion.
is_related_to	Threshold	AlternativeValue	A threshold can be related to one or two alternative values. Each alternative can have at most three thresholds.
leads_to	DMSituation	DMObject	Each DM situation leads to a DM object. A DM object can be related to several DM situations.
makes	DecisionMaker	IntuitiveDecision	A Decision-maker can make intuitive decisions. An intuitive decision is made by a decision-maker.
responds_to	Decision	DMSituation	A decision responds to a DM situation. A DM situation can be related to none or several decisions.
validates	DecisionMaker	Decision	A decision-maker can validate decisions.

In the following section, we resume the DM ontology and explain it through an example.

1.4. Illustration with ERP

We give some additional explanations within a common example, which is a project portfolio management (PPM), for instance a project of an ERP tool purchase (called the ERP project).

The starting point for analyzing the DM ontology is the *DM situation*. The DM situation is an abstract concept that puts together the main DM elements and describes a concrete case of DM dealing with a given *DM object*. The DM object can be a *process* or a *product* element. In the case of PPM, the `ERP project` is the DM object, which is a `product` element.

A given DM situation contains a *DM problem* and a *set of alternatives*. In PPM, the problem is a `choice` of a relevant project; alternatives are `global` as they exclude each other and `true` as they are all valid for comparing. The firstly identified set of potential ERP cannot evolve quickly, thus the alternative set is `stable`. Alternatives can be the `SAP ERP` project, the `Oracle e-Business Suite` project, and the `OpenERP` project. The DM situation can also contain a *set of criteria*. It is not mandatory as a decision could be made without analyzing criteria. Criteria can have different

natures. It can be: (i) intrinsic characteristics of alternatives (the *characterizes* relationship), (ii) future consequences of alternatives depending on the future state (the *is(3)* relationship), (iii) decision-makers' goals (the *is(2)* relationship), or (iv) decision-makers themselves (in this case, they have a role of *stakeholders* according to the *is(1)* relationship). In the PPM case, two criteria can be considered: `purchase cost` and `maintenance cost`. The first one is known directly and constitutes a `characteristic` of a project. The second one depends on several factors (for instance, project duration) and implies the `consequences` of the ERP implementation in the future.

The criteria could be analyzed in order to know their *weights*, *preference rules*, and *thresholds*. In our case, weights could be `equal` (same importance of two criteria); the preference rule is the `minimization` of two costs; a threshold could be established in order to indicate the `maximal acceptable cost` of the ERP purchase and maintenance.

All alternatives are evaluated according to the identified criteria in order to obtain the *alternative values*. In the ERP purchase case, a value matrix (3×2) will be constituted. The alternative values can be aggregated in order to produce a unique value by alternative (the *aggregates (1)* relationship). For instance, it could be a weighted sum. In the PPM case, the two ERP costs will be `added` for each alternative ERP. Thus, each ERP will have only one value allowing comparing all ERP in an easier way. Based on these aggregated values, a `method-based decision` will be made. The obtained decision is the selection of an alternative.

Decision-makers can participate in DM by several ways. They *define* the DM problem; *have goals* and *preferences* with regard to preference rules, weights and thresholds. They can also become criteria as a particular decision-maker type – *stakeholder*. Decision-makers evaluate *alternatives*, *validate* decisions and *make intuitive decisions*. In our case, a decision-maker participates in the `definition` of the DM problem, in the `establishment` of weights, preference rules, and thresholds. He also `validates` the final decision. The given decision-maker is an `individual` stakeholder.

Both method-based and intuitive decisions *are related to* the DM situation. Each DM situation can lead to either none or several decisions.

The application of DMO to the ERP Project case is illustrated at Figure II.1.22.

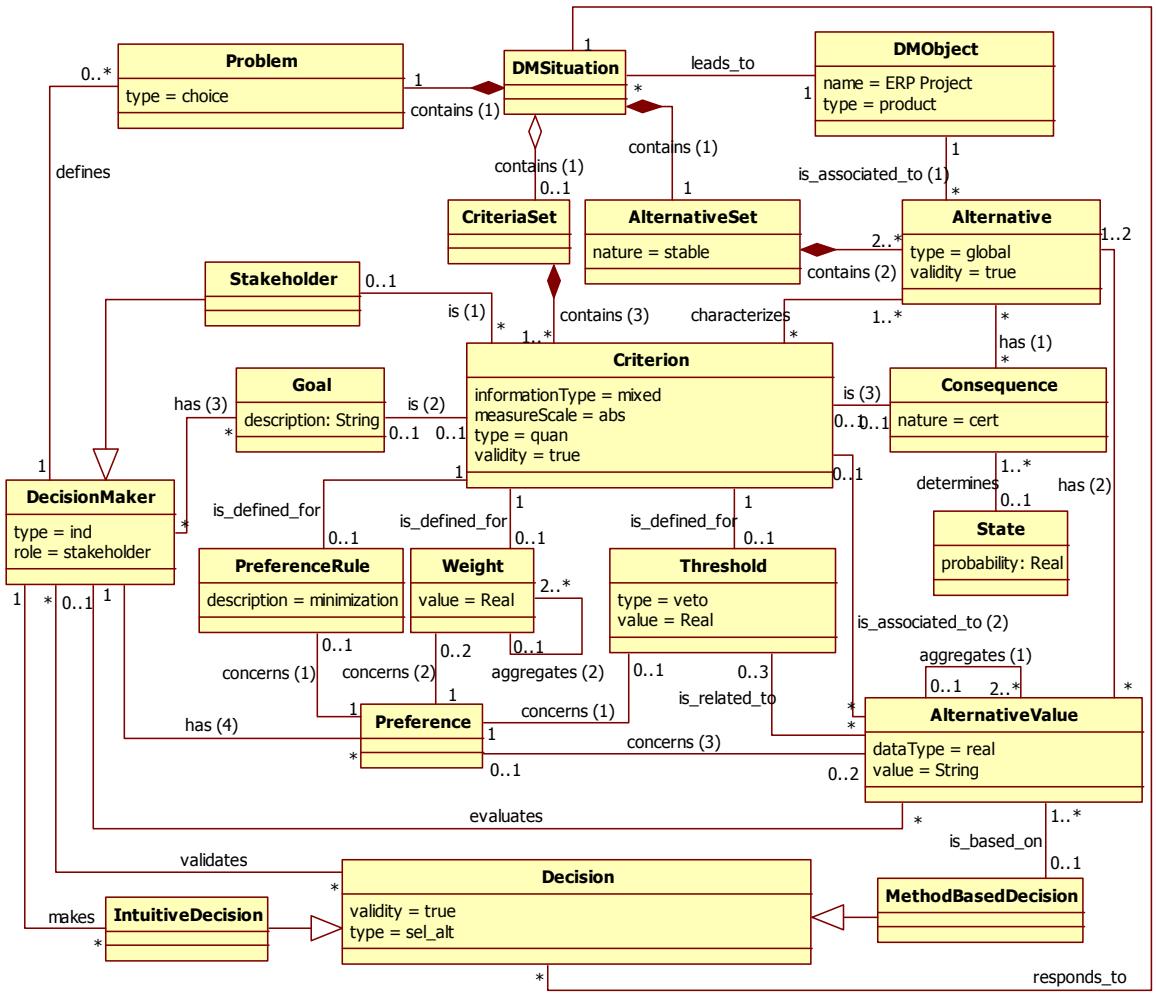


Figure II.1.22. DMO: Application to the ERP Project Case.

1.5. Application Case: the REDEPEND Approach

This section aims at validating the DM ontology. Our goal is to show how existing DM models could be expressed through the DM ontology. We have chosen an existing and well-known DM method dealing with requirements engineering: the REDEPEND approach [Maiden *et al.*, 2002]. We capture its DM model and express it through DMO, i.e. DMO concepts, attributes and relationships are used in order to represent the REDEPEND approach.

The REDEPEND approach uses requirements for selecting candidate tasks, which represent possible alternatives. It is based on the AHP (Analytic Hierarchy Process) DM method. The AHP, proposed by T.L. Saaty [Saaty, 1980], includes the pair-wise comparison between alternatives and/or criteria and the aggregation of the comparison results into quantitative indicators (score). The REDEPEND approach integrates the AHP and *i**, which is a well-known requirements modeling formalism.

Figure II.1.23 presents the DM concepts used in the REDEPEND approach. The DM situation in this approach is characterized as follows. The DM problem is *ranking*. The *DM object* is a task, which can be a scenario (*process*) or a goal (*product*). Tasks represent *alternatives*, which are *fragmented*, i.e. they can be dependent one another. All alternatives are *true* as the REDEPEND approach does not contain a module for validating them. The alternative set is *evolving* as it can change through time. One or more decision-makers (*individual stakeholders*) define goals and soft goals. Goals and soft goals represent requirements and are considered as *criteria* in the given model. These goals are *determinist*; their measure scale is *nominative*; the data type is *qualitative*; and they are *valid*.

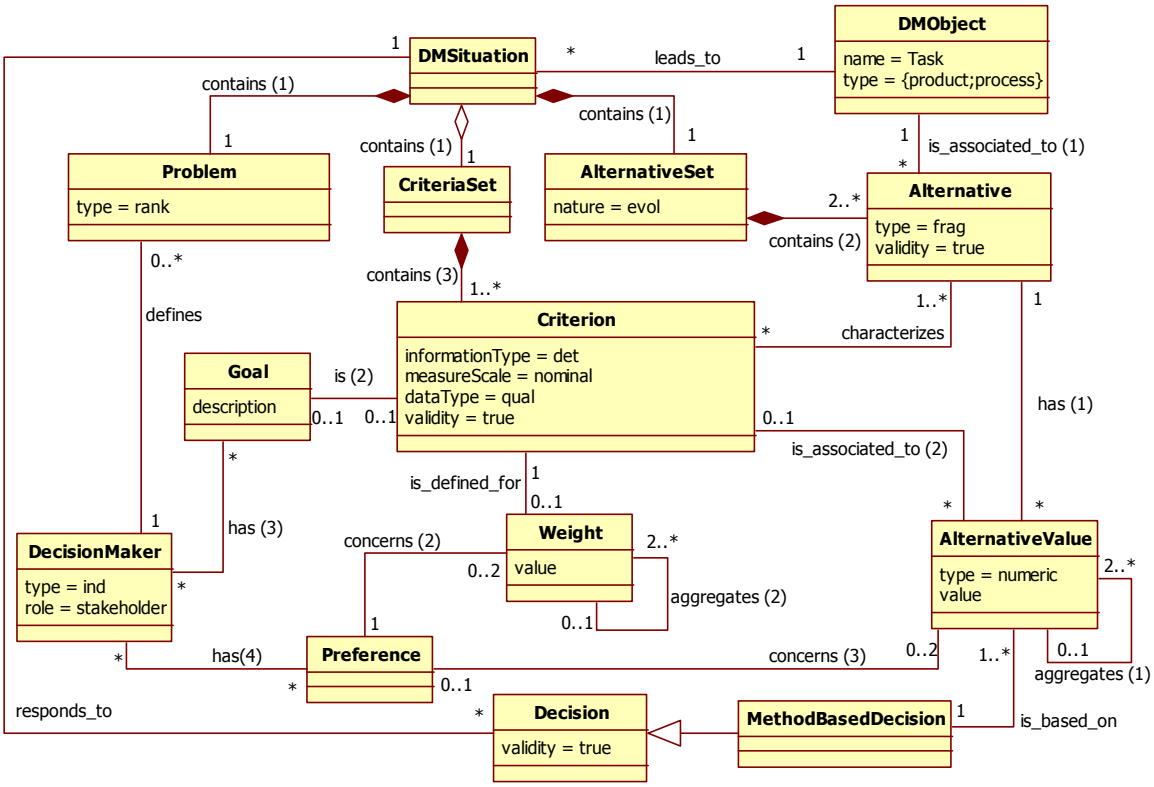


Figure II.1.23. DMO: Application to the REDEPEND Approach.

Decision-makers make pair-wise comparisons of tasks and goals. In this way, they express preferences on weights and alternatives values (the *concerns (3)* relationships). For instance, they compare each pair of alternatives according to a criterion and give a *numeric value* to it. A value can vary from 1 (equal importance) to 9 (absolute importance) in accordance with the basic AHP method. This constitutes the elementary *alternative value*. These values for all alternatives are then aggregated in order to rank the alternatives against the given criterion (*aggregates* relationship). The same analysis is made between alternatives for each criterion and between criteria in order to prioritize them too (the *weight* class and the *aggregates (2)* relationship respectively). The ranked alternatives and criteria are computed for the final alternatives' ranking in order to make decision (*is_based_on* relationship).

1.6. Conclusion

In this chapter, we have presented the DM ontology that models and formalizes DM knowledge. The DM ontology has been designed based on the state-of-the-art on ontology construction in the ISE field. It includes concepts, their properties and relationships. We have also shown that existing DM models can be expressed through DMO by applying it to a well-known DM method from the requirements engineering field.

The main goal of DMO is to enhance and facilitate DM. It supports IS engineers in DM activities as it includes all necessary elements and links for supporting DM in various situations. In our approach, the DMO will be used in two ways: (i) to formalize DM methods and method components in a standardized manner and (ii) to help formalizing a given DM situation for the further selection of appropriate DM components.

Part III. MADISE Decision-Making Method Family

The DM Ontology defined in Part 2 allows us to compare and use all DM methods on the same base. However, DM methods may be complex and difficult to use integrally. They are not adapted to the project at hand but defined in a general way. One of our goals is to be able to use DM methods following a concrete project situation. We consider that the situational method engineering (SME) field proposes techniques that will allow us to adapt existing DM methods or to create new DM methods with parts of the existing ones. Chapter 1 offers a state of the art of the SME approaches in order to identify these techniques, their advantages and their drawbacks.

Most of the SME approaches use the concept of modularity with the decomposition of methods into components that are then assembled to construct new methods. We define in Chapter 2 the notion of component for the DM field. The representation and organisation of these components is also addressed with the definition of the DM method family. The concept of method family is based on two existing meta-models: COMET and MAP.

One of the underlying notions used in SME is the context-awareness. However, this notion is never formally defined. Chapter 3 proposes a context typology, specific for the DM domain, which can be applied on the DM method family. An illustration of this contextualization is proposed for a specific case.

In practice, once the project context has been defined, several possibilities are offered to the IS engineer. The navigation through the DM family, in order to define the concrete IS DM process that will be used on the project, may be executed in different ways. Chapter 4 addresses this issue with the definition of several family configuration possibilities.

Chapter 1: Analyzing SME Approaches

1.1. Introduction

We postulate in this thesis that the method engineering domain may offer techniques to improve the application of DM methods in the IS field. Method Engineering is the discipline to design, construct and adapt methods, techniques and tools for the development of IS [Brinkkemper1996]. ME approaches, and in particular, situational ME approaches, highlight some very interesting aspects to improve the use of DM methods: modularity, reuse, flexibility, and situation-awareness.

- First of all, they consider the methods as *modular*, which means that a method may be divided into blocks that can be recomposed together following the needs of the project at hand.
- This decomposition helps to improve the *reuse* of methods, as each block may be used several times. The reuse principle allows constructing methods based on existing methods or method blocks conceived earlier.
- This combination of modularity and reuse aims to give more *flexibility* to the method engineer to construct methods adapted to a specific project. According to [Harmsen *et al.*, 1994] the flexibility can vary from rigid methods to modular methods.
- This is the reason why ME approaches are considered as adapted to concrete situations and are called *situation-aware.*

The purpose of this Chapter is to present different approaches of Situational Method Engineering by analyzing them according to the framework elaborated for this purpose. After a brief introduction to the ME science, we describe the proposed four views framework. Then, we offer a brief description of the main SME approaches. And, finally, we apply the suggested framework to these approaches in order to identify drawbacks of the existing SME approaches with reference to the DM field.

1.2. Method Engineering Fundamentals

The diversification of IS methods and their increasing complexity put forward a lot of requirements to engineering methods. They must accept standardization, manage complexity, but also have to be flexible enough to anticipate new types of applications. Some have been proposed to cover new application areas. However, to define a comprehensive method is a lengthy process and methods are often already outdated when published. In addition, they can never fully be applied to an IS engineering project because each project is different and the methods are, by definition, rigid and standardized. Each information system is developed for a specific situation and each situation is different. There is today no method that can be used in any situation. On the one hand, a method must accomplish standardization and, on the other hand, it must remain flexible to be adapted to the situation.

It should be noted that traditional analytical methods (like UML [UML, 2011]) were not built to be adapted to a situation in progress, but in order to resolve the problems found in current IS. They were developed to solve problems in some typical situations. Unfortunately, IS are often very specific and include items that are not taken into account by these methods. For all these reasons, current methods do not allow users to be guided effectively in their work or to share and reuse their expertise in a systematic way. We must find a way to integrate consideration of these difficult issues in the original method (in the product part as well in the process part). The method engineering (ME) purpose is to address these issues.

1.2.1. Method Engineering

Method Engineering aims to bring effective solutions to the construction, improvement and modification of methods used to develop information and software systems. One of the goals of ME is to improve the use of methods. ME proposes an adaptation framework where methods are created to match specific organizational situations. This is often done at two levels. At a global level, it determines the contingency factors of the project [Van Slooten *et al.*, 1996] helping to select the correct method to use. At a finer level, it manages the construction process for the current situation [Deneckère, 2001].

A method must perform standardization, but it must remain flexible to take into account specific situations. Harmsen [Harmsen, 1994] uses the term of *controlled* flexibility to set this requirement. This flexibility is achieved in most approaches of engineering methods, by defining a preliminary stage in the project to characterize the status of the project and then, using this characterization, to compose an appropriate method to deal with this situation. For this reason, several authors tried to design methods that would be as effective and as adapted as possible to the needs of the information systems development [Firesmith et al., 2001] [Rolland et al., 1992]. This goal was not always reached, especially because the methods were not always well adapted to projects specificities.

Method engineering has seen the development of a new group of methods called *situational* methods that allow building and adapting methods to specific projects. The situational approach finds its justification in the practical field analysis, which shows that a method is never followed literally [Ralyté 2001] [Mirbel et al., 2002]. The discipline of Situational Method Engineering (SME) promotes the idea of retrieving, adapting and tailoring components, rather than complete methodologies, to specific situations [Rolland et al., 2001b]. In order to succeed in creating methodologies that best suit given situations, components (building blocks of methodologies) representation and cataloguing are very important activities.

1.2.2. Basic Definitions

The basic notions of the SME science are method, product, and process; all studied in the following sub-sections.

1.2.2.1. Method

The origin of the word *method* is the Greek word "methodos" (pursuit of knowledge), from "meta" (expressing development) and "hodos" (way). Among all the definitions of a method proposed in [Olle *et al.*, 91] [Brinkkemper, 1990] [Lyytinen *et al.*, 1989] [Kronkof, 1993] [Prakash, 1994] [Prakash, 1999] [Seligmann *et al.*, 1989] [Harmsen, 1994], we retain the fact that most of them converge on the following statement:

> *A method is based on one or several product models (systems of concepts) and consists of one or more process models (steps executed in a given order).*

More precisely, [Brinkempper, 1996] offers the following definition:

> *A method is an approach to perform a system development project, based on a specific way of thinking, consisting of directions and rules, structured in a systematic way in development activities with corresponding development products.*

We can therefore say that a method provides the concepts to describe the product and the methodological rules to perform the process to produce this product with reasonable efficiency. The Product part is the desired target of an IS development while the Process part is the way forward to reach this target.

1.2.2.2. Product

A product is the result of the application of a method [Rolland, 2009]. The Product part is represented by different models. A *product model* is the notation used to describe the product. A UML class diagram is an example of notation. The product part results from the application of the process associated with it. A method can have multiple *product models* corresponding to different notations [Olle *et al.*, 91]. The product model is composed of a set of concepts and rules to manipulate them to model the Product part of a method.

1.2.2.3. Process

The Process part represents the process to be carried out to define the product part. It represents a coherent set of activities to build an Information System. It is composed of all the steps and heuristics to specify the decisions, how to take them and in which order. An *approach* or *process* is, in most cases, "a set of interrelated activities performed in order to define a product [Franckson et al., 1991] [Deneckère, 2001]. A *process model* describes the way of organizing the production of the product: the steps and actions to develop an Information System. In the community of IS, a process model corresponds to the methodological approach prescribed by the used method.

1.2.3. Principles of Method Engineering

The ME aims at responding to some main requirements for Information System Developments (ISD) processes: being situational, variable, flexible, and reusable. In order to meet these needs, ME follows some principles.

In [Rolland, 2005] [Rolland, 2009], four ME principles are defined:

- Meta-modelling,
- Reuse,
- Modularity,
- Flexibility.

The *meta-modelling* allows representing method models. By meta-modelling, [Rolland, 2009] means the modelling of models that make up a method. Commonly, a method meta-model contains two elements: a meta-model of product and a meta-model of process, both elements are interconnected. The meta-models are used for method constructing, formalizing, comparing, standardizing, and defining the links between engineering methods and programming languages. On the same way, [Prakash *et al.*, 2002] proposes to use a generic layer to help in instantiating process meta-models or integrated process-product meta-models of diverse domains. This genericity can be viewed as replacing meta-model dependence with generic model dependence.

The *reuse* allows constructing methods on the basis of existing methods or method components conceived earlier. This principle is used in order to build new ISD methods by assembling various experienced method parts viewed as method components [Rolland, 2009].

The *modularity* is necessary to assure reuse and flexibility of methods. According to this principle, a method is considered as a set of reusable components, which can be applied separately. The notion of method component is central to SME as it encapsulates the method knowledge. There are various representations of modular parts: fragments [Brinkkemper, 1996], chunks [Rolland *et al.*, 1998], components [Wistrand *et al.*, 2004], OPF fragments [Henderson-Sellers, 2002] and method services [Rolland, 2009] [Guzélian *et al.*, 2007] [Iacovelli *et al.*, 2008]. In addition, a particular kind of the knowledge representation is method pattern [Deneckère, 2001] [Deneckère *et al.*, 2001] which represents an executable component aiming at extending an existing method by new features.

The *flexibility* is needed to adapt methods to different projects. According to [Harmsen et al., 1994] the flexibility can vary from rigid methods to modular methods (between these two extremities are found selection from rigid methods, paths within one method, selection and tuning of a method outline). Different factors may lead to adaptation: project contingencies, specific requirements of a used group and dynamic changes in the ISD process [Rolland, 2009].

1.3. Framework for Comparing SME Approaches

The SME science has induced the development of many different approaches allowing creating new methods depending on characteristics of a given project. In order to compare them, we have elaborated a four views framework of SME approaches.

The four views framework, originally proposed in [Jarke et al., 1992], has proved its efficiency in enhancing the understanding of various engineering disciplines such as information systems engineering [Jarke et al., 1992], requirements engineering [Jarke et al., 1993], IS development process engineering [Rolland1998] and method engineering [Rolland 1997].

In the original framework [Jarke et al., 1992], the views where described as follows:

- The ***subject view*** contains knowledge of the domain about which the proposed IS has to provide information. It contains real-world objects, which become the subject matter for system modeling.
- The ***system view*** includes specifications of what the system does, at different levels of detail. It holds the modeled entities, events, processes, etc. of the subject world as well as the mapping onto design specifications and implementations.
- The ***usage view*** describes the organizational environment of the information system, i.e. the goals and the activity of agents and how the system is used to achieve work, including the stakeholders who are system owners and users.
- The ***development view*** focuses on the entities and activities, which arise as part of the engineering process itself.

Our point of view is that this framework concept can be used to help in understanding and comparing different SME approaches.

For our purpose, we define the SME four-dimensional framework as follows:

- **Usage view.** In the usage view, the corresponding dimension allows investigating the rationale of SME approaches (the *Why*).
- **Subject view.** This view expresses the dimension, which deals with the nature of SME approaches (the *What*).
- **System view.** The system view answers the question about the representation used by the SME approaches (the *Which*).

- **Development view.** This view deals with different aspects that describe the usage of SME approaches (the *How*).

The four views composing the four-dimensional framework try to answer the following questions about situational methods:

- Why is the SME approach?
- What is the SME approach?
- Which is the representation of the SME approach?
- How is the SME approach used?

This allows us to discuss in a focused manner the different concerns of SME: the reasons of SME approaches, their representations, the way of developing these representations, and, finally, the way of using these representations. This is done in the system, subject, development, and usage views respectively (See Figure II.1.1.).

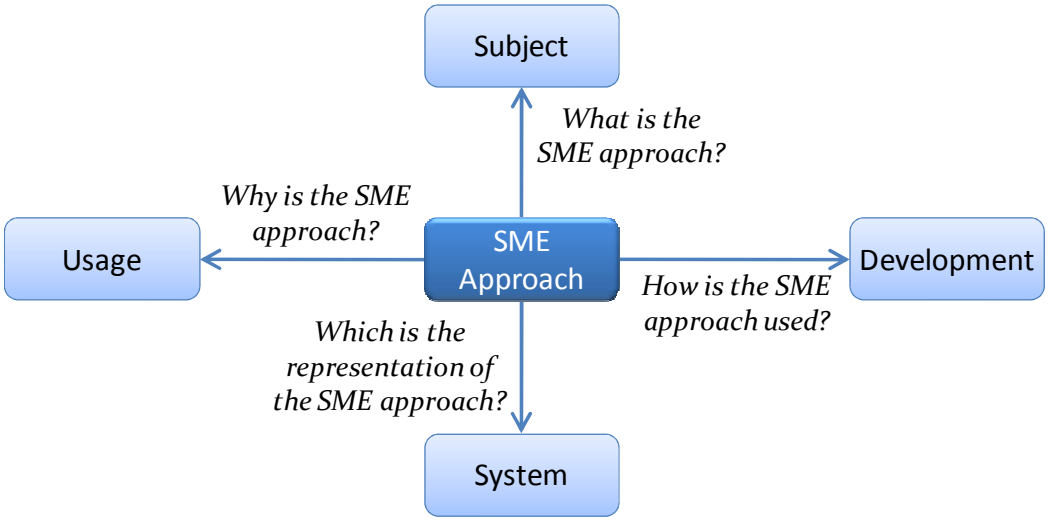


Figure II.1.1. Four Views for SME Approaches.

Each view is described by a set of attributes that allow a more detailed study of the situational methods. As a result, our framework contains 14 attributes organized into four views developed in the following subsections.

1.3.1. Usage View

This view captures why we should use a specific SME approach and what are the issues allowed by its practical application. The objective view contains two attributes: Covered way and Target issues represented at Figure II.1.2 and detailed in the corresponding sub-sections.

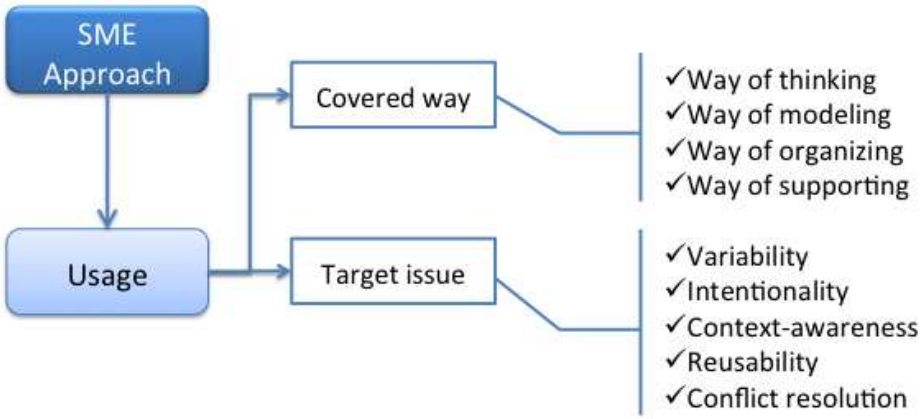


Figure II.1.2. Usage View Representation.

1.3.1.1. Covered Way

Seligmann studies the definition of methods and gives a definition of a method as "a way of thinking, a way of modeling, a way of organizing and a way of supporting" [Seligmann *et al.*, 1989]. However, even if a lot of fragments are considered as complete method, often they are not adapted to satisfy all these requirements. We investigate this question through the *covered way*.

- The *way of thinking* refers to the philosophy of conceiving methods.
- The *way of modeling* shows the models to be constructed within the method application.
- The *way of organizing* includes the way of working (the means to perform the development) and the way of control (the manner to manage the development).
- The *way of supporting* deals with the description techniques and corresponding tools [Rolland 1997].

The covered way can be represented as follows:

```
Covered way: ENUM {way of thinking, way of modeling, way of
organizing, way of supporting}
```

1.3.1.2. Target Issue

We investigate the SME approaches in order to identify whether they are relevant to the issues established for our proposal. For this reason, we analyze the existing approaches in order to check whether they consider the *target issues* such as variability, intentionality, context-awareness, reusability, and conflict resolution.

- The *variability* issue shows whether a SME approach considers the variability in an explicit way.
- The *intentionality* issue shows whether intentions or actors' goals are considered by a SME approach.
- The *context-awareness* issue allows considering the situation in a given SME approach.

- The *reusability* issue indicates whether a SME approach enables to reuse the same methods or their components in different situations.
- The *conflict resolution* issue shows if a given SME approach allows obtaining an agreement between divergent and often contradictory points of view of different actors.

We represent this attribute as follows:

```
Target issue: ENUM {variability, intentionality, context-awareness,
reusability, conflict resolution}
```

1.3.2. Subject View

The Subject view deals with the nature of SME approaches. Three facets allow covering this aspect: Knowledge construction, Knowledge organization, and Construction flexibility. Figure II.1.5 shows these attributes, detailed in the following sub-sections.

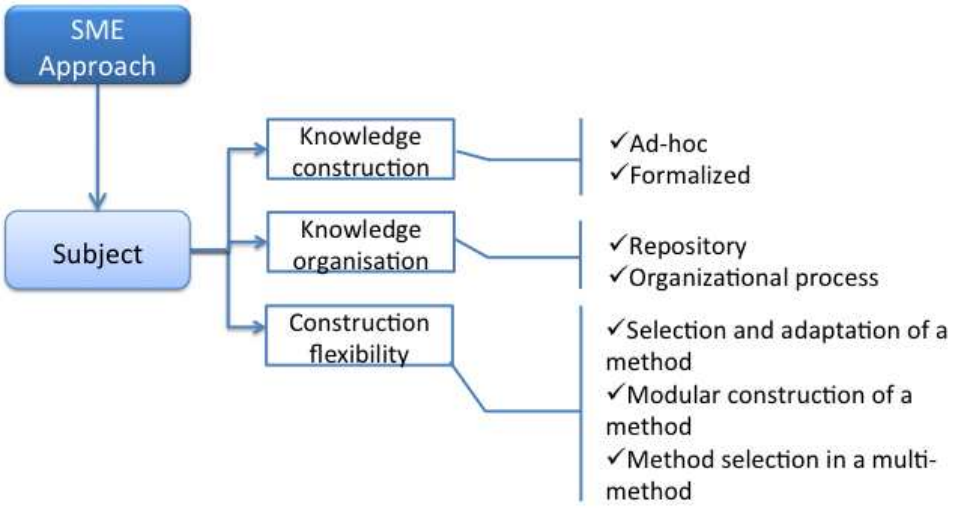


Figure II.1.5. Subject View Representation.

1.3.2.1. Knowledge Construction

In the traditional way, knowledge is constructed is by engineer experience. As long as this experience is not formalized and that basic available knowledge does not constitute an available part for the various applications, one can say that this knowledge is the result of an ad-hoc construction technique [Bajec, 2005]. This has two major consequences: ignorance on the way in how was carried out the construction and dependence on the field of expertise. If this knowledge must be independent of the experience field and rapid to built, it is then necessary for construction techniques based on the experiment to use more formalized techniques [Karlsson *et al.*, 2004]. In this manner, we distinguish two ways of knowledge construction: ad-hoc and formalized.

- The *ad-hoc* knowledge construction is based mainly on the experience and expertise of engineers.
- The *formalized* knowledge construction is based on a set of predefined techniques which support the construction process.

We represent this attribute as follows:

```
Knowledge construction: ENUM {formalized, ad hoc}
```

1.3.2.2. Knowledge Organization

The knowledge used during SME construction can be stored in a library or repository to be reused later. They provide the basic functions for the management of a components repository. As these libraries can contain a large number of components, they generally offer research techniques, as indexation techniques or the use of keywords. Other approaches, in addition to the component extraction formalism, have an organizational process, which helps to manage the knowledge consistency. The organization processes thus allow managing this problem in a more formal way [Rolland et al., 2004] [Karlsson *et al.*, 2004].

The knowledge organization can be described as follows:

```
Knowledge organization: ENUM {repository, organizational process}
```

1.3.2.3. Construction Flexibility

[Harmsen *et al.*, 1994] proposed a spectrum to organize the engineering method approaches according to their degree of flexibility towards a new situation. The methods are organized on a scale of flexibility varying from “low” to “high”. At the “low flexibility” level are the rigid methods while, at the “high flexibility” level, we find the SME methods. It includes: use of “rigid” methods, selection from rigid methods, toolkit/multiview approach, paths within one method, selection and tuning of a method outline, and modular method construction [Harmsen, 1997]. Based on this typology, we distinguish the following values of the construction flexibility attribute: selection and adaptation of a method, modular construction of a method, and method selection in a multi-method. A multi-method is a group of methods having the same purposes.

- *Selection and adaptation of a method*. This approach allows selecting an existing method and adapting it following the project’s needs.
- *Modular construction of a method*. This approach enables creating a new method from different components retrieved in the method repository according to the project specificities.
- *Method selection in a multi-method*. This approach allows selecting a path from a multi-method, which in turn represents a set of methods of the same domain unified by their common elements.

The construction flexibility is expressed in the following manner:

```
Construction flexibility: ENUM {selection and adaptation of a
method, modular construction of a method, method selection in a
multi-method}
```

1.3.3. System View

The System view answers the question about the representation used by the SME approaches. This view includes six attributes: Knowledge genericity, Knowledge representation, Actor representation, Variability explicit representation, Context representation, and Abstraction level. This view is summed up at Figure II.1.3.

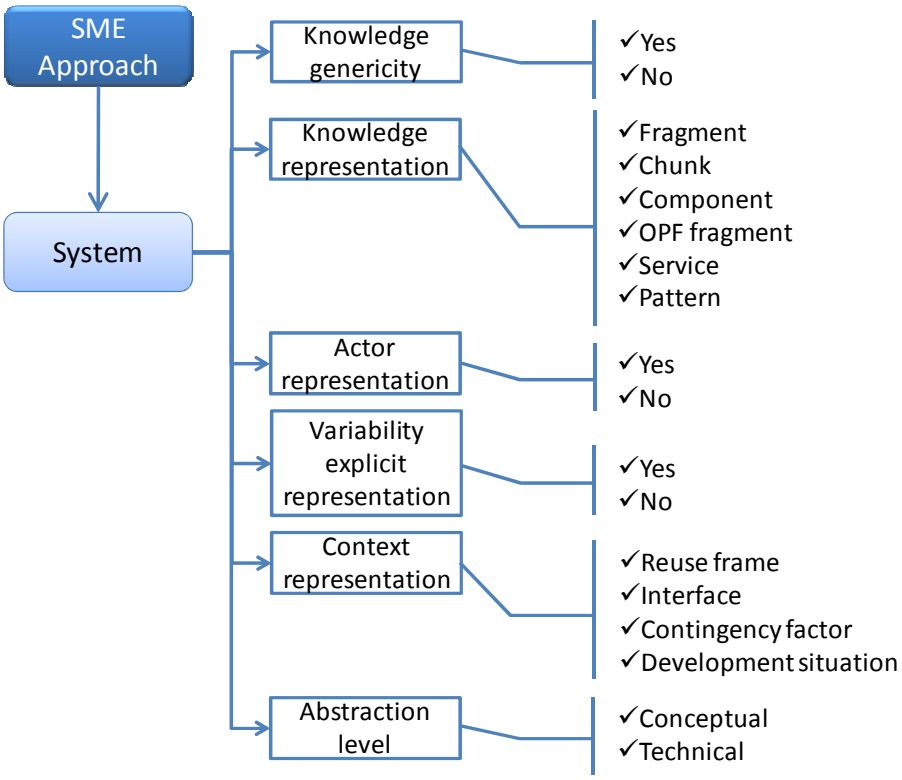


Figure II.1.3. System View Representation.

1.3.3.1. Knowledge Genericity

Despite some standardisation efforts of the ME community, all approaches are strongly coupled with their own notion of method component so the techniques developed in one approach are not usable in another one [Rolland, 2009]. In order to consider this aspect, we suggest the *knowledge genericity* attribute, which indicates whether a SME approach is dependent of the knowledge representation (of the method components type). We distinguish two possible values: Yes and No.

- *Yes*, if a SME approach is strongly related to the type of knowledge representation.
- *No*, if a SME approach does not depend on the knowledge representation type.

The knowledge genericity is resumed in the following manner:

```
Knowledge genericity: ENUM {Yes, No}
```

1.3.3.2. Knowledge Representation

The notion of method component is central to SME as it encapsulates the knowledge in a given SME approach. There are various representations of modular parts: fragments [Brinkkemper, 1996] [Harmsen *et al.*, 1994], chunks [Rolland *et al.*, 1998], components [Wistrand *et al.*, 2004], OPF fragments [Henderson-Sellers, 2002], method services [Rolland, 2009] [Guzélian *et al.*, 2007] [Iacovelli *et al.*, 2008], and method patterns [Deneckère, 2001] [Deneckère *et al.*, 2001].

- ***Method fragment*** [Brinkkemper1996] [Harmsen *et al.*, 1994]. Fragments are standardized building blocks based on a coherent part of method. A fragment is either a Product or a Process fragment and is stored in a method base from which they can be retrieved to construct a new method following assembly rules [Brinkkemper *et al.*, 2001].
- ***Method chunk*** [Rolland et al., 1998] [Ralité *et al.*, 2003]. A chunk is described as a way to capture more of the situational aspects in ME and to appropriately support the retrieval process. It concentrates on the intention of the method component and on the situation in which it may be applied (the interface). Contrary to the preceding representations, a chunk embeds at the same time the product and process parts.
- ***Method component*** [Wistrand *et al.*, 2004] [Agerfalk, 2003] [Karlsson, 2005]. Components allow viewing methods as constituted by exchangeable and reusable components. Components are unified into configuration templates and configuration packages.
- ***OPF fragment*** [Henderson-Sellers, 2002]. In the OPEN Process Framework (OPF), the fragment is generated from an element in a prescribed underpinning meta-model. This meta-model has been upgraded with the availability of the international standard ISO/IEC 24744 [ISO/IEC 24744, 2007].
- ***Method service*** [Guzélian *et al.*, 2007] [Rolland, 2009] [Iacovelli *et al.*, 2008]. The method service represents a chunk from the structural point of view, but it is realized using the Service Oriented Architecture (SOA).
- ***Method pattern*** [Deneckère, 2001] [Deneckère *et al.*, 2001]. A method pattern represents an executable component aiming at extending an existing method by new features. The key idea concerning patterns is the fact that a pattern relates a *problem* to its *solution*.

The knowledge representation can be classified as follows:

```
Knowledge representation: ENUM {fragment, chunk, component, OPF
fragment, service, pattern}
```

1.3.3.3. Actor Representation

Actors are not always viewed in the same manner in different SME approaches. Their participation is often tacit. However, several approaches include actors in their meta-models in order to indicate

roles that actors can play in method engineering processes. For this reason, we suggest the *actor representation* attribute, which can take the following values: yes and no.

- *Yes*, if an approach considers actors' roles,
- *No*, if an approach ignores the concept of actor.

The actor representation is following:

```
Actor representation: ENUM {yes, no}
```

1.3.3.4. Variability Explicit Representation

The notion of variability is defined as the ability of a software system to be changed, customized or configured to a specific context [Van Gurp, 2000]. This notion is also used in the SME science, as different method components are stored in a repository and used in a given project in function of the situation characterization. However, the variability is nearly never explicitly defined in the approaches. The corresponding attribute *Variability explicit representation* has two possible values: yes and no.

- *Yes*, if an approach explicitly includes the variability representation,
- *No*, if an approach does not explicitly represent the variability.

The variability is represented as follows:

```
Variability explicit representation: ENUM {yes, no}
```

1.3.3.5. Context Representation

This attribute captures different representations of context in the SME approaches. Based on the reviewed literature, the following representations are identified: reuse frame, interface, contingency factor, and development situation.

- **Reuse frame.** The reuse frame [Mirbel, 2008] is a framework representing different factors that affect IS development projects. These factors are called criteria. Reuse frame allows specifying a context of method components reuse, searching method components and comparing between them in order to find an alternative component to a used one. The reuse frame model includes a reuse situation (which is a set of criteria classified into three dimensions: organizational, technique and human) and reuse intention.
- **Interface.** In [Ralyté *et al.*, 2001], the method chunk context is defined by its interface that includes a situation and an intention. The situation represents the conditions in which the method chunk can be applied in terms of required inputs product(s). The intention is a goal that the method chunk helps to achieve. Therefore, the interface model includes two elements: the situation and the intention. These two first approaches have been unified in [Mirbel *et al.*, 2006]. The method service approach [Guzélian *et al.*, 2007] reuses the notion of interface as it aims at describing the situation in project development for which the method service is suitable and defining the purpose of the service. Its model includes domain

characteristics (project nature, project domain) and human (actor), process and product ontologies.

- **Contingency factors.** Situations (the context) are described by a set of characteristics called contingency factors [Van Slooten *et al.*, 1996] or project factors [Harmsen, 1997] [Harmsen *et al.*, 1994]. These factors are used to define the project situation by assigning values to them. In [Van Slooten *et al.*, 1996], four categories are given: domain characteristics (describing the content of the system), external factors (laws and norms), technical factors (related to the development platform) and human factors (representing the development expertise of people).
- **Development situation.** [Karlsson *et al.*, 2004] defines the development situation as an abstraction of one or more existing/future software development projects with common characteristics. This situation is used to characterize the specific projects and to select configuration packages (method fragments). The development situation model includes a characteristics set.

We represent the context representation attribute as follows:

```
Context representation: ENUM {reuse frame, interface, contingency
factor, development situation}
```

1.3.3.6. Abstraction Level

[Brinkkemper *et al.*, 2001] explores the notion of the component formalism that can be either conceptual when components are expressed by descriptions and specifications of methodology parts, or technical when there is an implementation of operational parts in tools. We apply this notion to the SME approaches in order to qualify them according to their abstraction level (conceptual or technical).

- *Conceptual*, as in [Karlsson *et al.*, 2004] where components are expressed by descriptions and specifications of methodology parts, or
- *Technical* as in [Ralyté, 2004] where there is an implementation of operational parts with tools.

The abstraction level can be described as follows:

```
Abstraction level: ENUM {conceptual, technical}
```

1.3.4. Development View

This view deals with different aspects that describe the situational methods construction. This implies to specify the following aspects: the used process model and construction technique and, if it exists, tools to help this development. In this manner, this view contains three attributes: Process model, Construction technique, and Tool/Implementation, represented at Figure II.1.6 and detailed in the corresponding sub-sections.

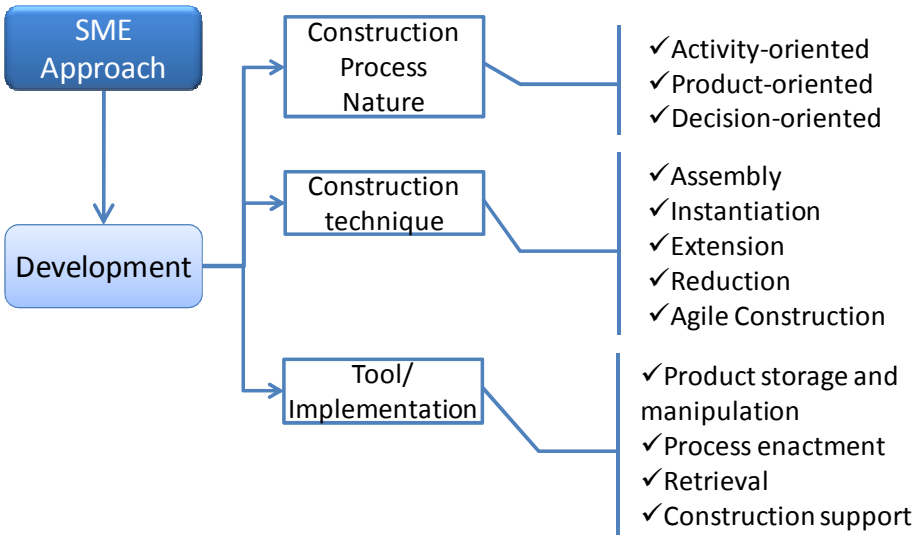


Figure II.1.6. Development View Representation.

1.3.4.1. Construction Process Nature

A process model prescribes a way of working to reach the desired target [Rolland, 2009]. Based on this, we define the nature of the construction process. This attribute is important as the process execution predefines the quality of the obtained product. According to Dowson [Dowson1988], process models can be classified into three groups of models called activity-oriented, product-oriented, and decision-oriented. [Rolland, 1998] refines this classification with a fourth group called context-oriented. Decision and context-oriented processes try to take decision-making into account as they integrate goals and arguments.

- *Activity-oriented* process models focus on the set and the order of activities that must be done in order to realize a product.
- *Product-oriented* process models have similarities with the state transition diagram [Rolland, 2009] as they link the state of the product with the corresponding activity.
- *Decision-oriented* process models consider the SME activities as a decision-making process and, consequently, as the successive transformations of the product based on the made decisions [Rolland, 2009].

We represent this attribute as follows:

```
Process model: ENUM {activity-oriented, product-oriented, decision-oriented}
```

1.3.4.2. Construction Technique

This facet represents the various ways of building a method. [Nehan *et al.*, 2007] distinguishes the following main manners to use fragments for constructing a new method according to project specificities: assembly, instantiation, extension, and reduction. In assembly-based approaches,

separate fragments are grouped with regard to the studied specific project to form a unique method [Ralyté *et al.,* 2003]. By applying extension, a basic method is transformed into a new one by addition of new fragments [Ralyté *et al.* 2003]. By reduction, some fragments are removed from the basic method in order to transform it to match the engineer's needs [Wistrand *et al.*, 2004]. In the real world projects, with time and resource constraints, where is a need for constructing methods dynamically depending on the project specificity and adapting it during its realization if project characteristics change. This property implies the agility of methods. Recently, the agility was discussed with regards to methods of IS development [Abrahamsson *et al,* 2002]. However, agility in ME approaches is not widely spread and is only suggested in recent works. To consider this kind of construction, we introduce the fifth value *agile construction* for the given facet. Thus, we distinguish the following main construction techniques: assembly, instantiation, extension, reduction, and agile construction.

- The *assembly* approaches concentrate on the grouping of method components belonging to complementary methods [Ralyté, 2001] [Reinhartz-Berger *et al.,* 2004]. They assemble separate selected method components with regard to the studied specific project to form a unique method.
- The *instantiation* approaches use an identification of the common and generic method characteristics and represent them by a system of concepts called meta-model. These approaches allow the creation of a whole set of methods sharing the same properties [Ralyté, 2001] [Rolland *et al.,* 2004] [Ralyté, 2004].
- The *extension* approaches allow the transformation of a basic method into a new method adapted to the project's needs [Deneckère, 2001] [Ralyté *et al.*, 2003] with addition of new functionalities in a base method.
- The *reduction* approaches allow the removal of basic method operators in order to transform it to match the engineer's needs [Karlsson *et al.*, 2004] [Bajec, 2005].
- The *agile construction* allows building methods dynamically depending on the project specificity and adapting it during its realization if the project characteristics change.

The construction technique attribute is defined as follows:

```
Construction technique: ENUM {assembly, instantiation, extension,
reduction, agile construction }
```

1.3.4.3. Tool/ Implementation

The method components application needs to be supported by a tool. [Brinkkemper, 1996] defines a tool as a "possibly automated mean to support a part of development process". We distinguish different ways to implement components: firstly, the implementation of process and product parts of the method component and, secondly, the implementation of components' storage, retrieval, and construction. Even if all ME approaches investigate storing methods components in the "method base" or "method repository" [Brinkkemper, 1996] [Rolland *et al.,* 1998] [Gonzales-Perez, 2007], this information is relevant as all the other implementation parts are founded on this one. Hence, our

tool/implementation attribute takes the following values: product storage and manipulation, process enactment, retrieval, and construction support.

- *Product storage and manipulation*. A SME approach allows storing method components in a repository.
- *Process enactment*. A SME approach implements the process that supports the method components execution.
- *Retrieval*. A SME approach enables retrieving the method components from the repository according to the project needs.
- *Construction support*. A SME approach contains a mechanism that allows constructing new methods based on components from the repository.

The Tool/ Implementation attribute is represented as follows:

```
Tool/ Implementation: ENUM {product storage and manipulation,
process enactment, retrieval, construction support}
```

1.4. SME Approaches' Review according to the Framework

We propose a review of seven SME approaches. We choose our method panel in the set of the most widespread approaches and with the intention to offer a more complete study of the different views and their facets.

1.4.1. Method Fragment Approach

The SME *fragment* based method consists in encouraging a global analysis of the projects while basing itself on contingency criteria [Brinkkemper, 1996] [Harmsen *et al.*, 1994]. *Method fragments* (cf. Figure 1.A) are standardized building blocks based on a coherent part of method. A fragment is either a Product or a Process fragment and is stored on a method base from which they can be retrieved to construct a new method following assembly rules [Brinkkemper *et al.*, 2001]. The projects and the situations are characterized by means of factors associated with the methods. [Van Slooten *et al.*, 1996] uses a contingency model based on 17 contingency factors which take value between Low and High as 'Importance of the Project', 'Knowledge and Experience', 'Stability' and so on. According to the authors, the characterization of the project allows them to select the method components appropriate to the project. Construction is supported by component assembly rules and constraints having to be satisfied by the created method.

Based on the comparison framework, the method fragment approach is presented in the following manner:

Usage View

```
Covered way = {Way of thinking, Way of modeling, Way of
    supporting}
```

Target issues = {Context-Awareness, Reusability}

Subject View

Knowledge construction = {Ad-hoc}

Knowledge organization = {Repository}

Construction flexibility = {Modular construction of a method}

System View

Knowledge genericity = {Yes}

Knowledge representation = {Fragment}

Actor representation = {No}

Variability explicit representation = {No}

Context representation = {Contingency factor}

Abstraction Level = {Conceptual, Technical}

Development View

Construction process nature = {Activity-oriented}

Construction technique = {Assembly}

Tool/ Implementation = {Product storage and manipulation, Retrieval, Construction support}

The detailed description of this approach can be found in [Brinkkemper, 1996] [Brinkkemper *et al.*, 2001] [Harmsen *et al.*, 1994] [Harmsen, 1997] [Wistrand *et al.*, 2004].

1.4.2. Method Chunk Approach

The assembly-based SME approach [Ralyté, 2001] [Rolland *et al.*, 1998] [Ralyté *et al.*, 2003] aims at constructing a method 'on the fly' in order to match as well as possible the situation of the project at hand. It consists of the selection of method components (called method chunks) from existing methods that satisfy some situational requirements and their assembly. This approach is requirements-driven, meaning that the method engineer must start by eliciting requirements for the method. Next, the method chunks matching these requirements can be retrieved from the method base. And, finally, the selected chunks are assembled in order to compose a new method or to enhance an existing one. Method chunks are composed of two parts: a product part which represents the structure of the product used in the component, and a process part which provides the necessary procedures to obtain the desired product.

According to the framework, the method chunk approach is described as follows:

Usage View

Covered way = {Way of thinking, Way of modeling, Way of supporting}

Target issues = {Intentionality, Context-Awareness, Reusability}

Subject View

Knowledge construction = {Formalized}

Knowledge organization = {Repository}

Construction flexibility = {Modular construction of a method}

System View

Knowledge genericity = {Yes}

Knowledge representation = {Chunk}

Actor representation = {No}

Variability explicit representation = {No}

Context representation = {Reuse frame, Interface}

Abstraction Level = {Conceptual}

Development View

Construction process nature = {Decision-oriented}

Construction technique = {Assembly, Extension}

Tool/ Implementation = {Product storage and manipulation, Retrieval, Construction support}

[Ralyté 2001] [Ralyté *et al.*, 2001] [Ralyté *et al.*, 2003] [Ralyté, 2004] [Mirbel *et al.*, 2006] [Mirbel, 2008] present the method chunk approach.

1.4.3. Method Configuration Approach

The Method configuration approach [Karlsson *et al.*, 2004] [Wistrand *et al.*, 2004] [Agerfalk, 2003] [Karlsson, 2005] offers a meta-method (called Method for Method Configuration) containing method components organized in configuration packages and configuration templates. These concepts are used to configure methods following the specificities of a project while creating reusable assets. Each time the configuration is carried out, it deals with only one method of the IS development. The reusability advantage is obvious since pre-made configurations can be used over and over again. Hence, there is no need to perform a complete method assembly or method configuration for each new project. Experiences can be gathered and reused more efficiently since they can be attributed to coherent set of prescribed actions common in the organization, rather than to context-free actions. This approach introduces the notion of method rationale which is the systematic treatment of the arguments and reasons behind a particular method [Karlsson, 2005] [Agerfalk, 2003]. In the same way, the component description contains its rationale.

Based on the comparison framework, the method configuration approach is characterized as follows:

Usage View

Covered way = {Way of thinking, Way of modeling, Way of organizing, Way of supporting}

Target issues = {Intentionality, Context-Awareness, Reusability}

Subject View

Knowledge construction = {Formalized}

Knowledge organization = {Repository, Organizational process}

Construction flexibility = {Selection and adaptation of a method}

System View

Knowledge genericity = {Yes}

Knowledge representation = {Component}

Actor representation = {Yes}

Variability explicit representation = {No}

Context representation = {Development situation}

Abstraction Level = {Conceptual}

Development View

Construction process nature = {Activity-oriented}

Construction technique = {Extension}

Tool/ Implementation = {Product storage and manipulation, Process enactment, Retrieval}

The method configuration approach is described in [Karlsson *et al.*, 2004] [Karlsson *et al.*, 2004b] [Karlsson, 2005] [Karlsson *et al.*, 2005].

1.4.4. OPEN Process Framework

The OPEN Process Framework (OPF) [Henderson-Sellers, 2002] uses a meta-model to generate method components that are stored in a repository. OPEN offers a set of construction guidelines that are considered to be part of existing methodologies used to construct new methods. The OPF meta-model is composed of five main meta-classes [Henderson-Sellers *et al.*, 2004] [Henderson-Sellers *et al.*, 2005]: Stage, Producers, Work Units, Work Products and Languages; a method component is produced as soon as a meta-class is instantiated. An OPEN guideline helps method engineers both to instantiate the meta-model element to create method components and to select the best method components (from the repository) in order to create the situational method.

Using the comparison framework, the OPF approach is characterized as follows:

Usage View

Covered way = {Way of thinking, Way of modeling, Way of supporting}

Target issues = {Context-Awareness, Reusability}

Subject View

Knowledge construction = {Formalized}

Knowledge organization = {Repository}

Construction flexibility = {Modular construction of a method}

System View

Knowledge genericity = {Yes}

Knowledge representation = {OPF fragment}

Actor representation = {Yes}

Variability explicit representation = {No}

Context representation = N/A

Abstraction Level = {Technical}

Development View

Construction process nature = {Activity-oriented}

Construction technique = {Assembly, Instantiation, Agile construction}

Tool/ Implementation = {Product storage and manipulation, Retrieval}

The detailed description of the OPF approach can be found in [Firesmith *et al.*, 2001] [Firesmith, 2006] [Henderson-Sellers, 2002] [Henderson-Sellers *et al.*, 2005] [Henderson-Sellers *et al.*, 2005b].

1.4.5. Method Service Approach

Method services are developed by two approaches: SO2M [Guzélian *et al.*, 2007] and MaaS [Rolland, 2009] [Iacovelli *et al.*, 2008]. Both use the Service-Oriented Architecture (SOA) in order to develop the concept of method service.

SO2M (Service Oriented Meta-Method) [Guzélian *et al.*, 2007] develops a kind of method fragments called *method services*. It offers a repository with a large variety of method fragments and a service composition process. During composition, the process guides developer's choices; it selects method services and delivers a method fragment that achieves a developer's requirement. The SO2M meta-model is based on three main principles: service orientation, task ontology for reuse of knowledge on development problems and dynamic construction of method services for generating tailored methods.

The MaaS (Method as a Service) approach is proposed by C. Rolland [Rolland, 2007] and is detailed in [Iacovelli *et al.*, 2008] [Deneckère *et al.* 2008] [Rolland, 2009]. This approach emphasizes the adaptation of Service Oriented Architecture (SOA) to method components by developing Method-Oriented Architecture (MOA). The method service contains two parts: descriptor and implementation parts. The *descriptor part* combines a semantic descriptor (based on the fragment definition of the method chunks approach) and an operational descriptor describing the *implementation part* that operates the process of the fragment. Technical issues of method services are addressed with the application of widely used standards of service-based approaches.

Based on the comparison framework, the method service approach is presented in the following manner:

Usage View

Covered way = {Way of thinking, Way of modeling, Way of supporting}

Target issues = {Intentionality, Context-Awareness, Reusability}

Subject View

Knowledge construction = {Formalized}

Knowledge organization = {Repository}

Construction flexibility = {Modular construction of a method}

System View

Knowledge genericity = {Yes}

Knowledge representation = {Service}

Actor representation = {No}

Variability explicit representation = {No}

Context representation = {Interface}

Abstraction Level = {Technical}

Development View

Construction process nature = {Decision-oriented}

Construction technique = {Assembly}

Tool/ Implementation = {Product storage and manipulation, Retrieval, Construction support}

This approach is presented in [Guzélian *et al.*, 2007] [Rolland, 2009] [Iacovelli *et al.*, 2008].

1.4.6. Method Extension Approach

The Method Extension approach [Deneckère, 2001] guides the method engineer by providing extension patterns that help identifying typical extension situations and provide advises to perform the required extension. This approach offers two ways to extend a method: 1) directly through the pattern-matching strategy or 2) by using some generic knowledge (meta-patterns) related to the domain for which the extension has to be done. The former help to match extension pattern stored in a library to the extension requirements whereas the latter select, first, a meta-pattern corresponding to the extension domain and, then, guides the method extension by applying the patterns suggested by the meta-pattern. Both ways-of-working use a library of extension patterns in different ways.

According to the framework, the method extension approach is described as follows:

Usage View

Covered way = {Way of thinking, Way of modeling, Way of organizing}

Target issues = {Intentionality, Context-Awareness, Reusability}

Subject View

Knowledge construction = {Formalized}

Knowledge organization = {Repository, Organizational process}

Construction flexibility = {Selection and adaptation of a method}

System View

Knowledge genericity = {Yes}

Knowledge representation = {Pattern}

Actor representation = {No}

Variability explicit representation = {No}

Context representation = {Interface}

Abstraction Level = {Conceptual}

Development View

Construction process nature = {Decision-oriented}

Construction technique = {Instantiation, Extension}

Tool/ Implementation = {Not exists}

[Deneckère, 2001] [Deneckère *et al.*, 2001] detail the method extension approach.

1.4.7. FIPA (Foundation for Intelligent Physical Agent) approach

The FIPA (Foundation for Intelligent Physical Agent) approach [Seidita *et al.*, 2004] [Cossentino *et al.*, 2005] [Cossentino *et al.*, 2006] [Cossentino *et al.*, 2007] [FIPA, 2003] entered the IEEE computer Society Standards Committee with the mission of promoting agent-based technologies and the interoperability of agents with other technologies. FIPA defines the method fragment with a process meta-model. In this model, a process is composed of a set of activities performed by some active entities whose task is to produce a well-defined state of an Artefact as input/output. A process is defined as strongly oriented to the production of products. As a result, a method component is defined as a reuse part of a design process composed of two elements: the structure of the product and the necessary procedures to construct this product [Cossentino *et al.*, 2006].

Based on the comparison framework, the FIPA approach is characterized as follows:

Usage View

Covered way = {Way of thinking, Way of modeling, Way of supporting}

Target issues = {Intentionality, Context-Awareness, Reusability}

Subject View

Knowledge construction = {Ad-hoc}

Knowledge organization = {Repository}

Construction flexibility = {Modular construction of a method}

System View

Knowledge genericity = {Yes}

Knowledge representation = {Fragment}

Actor representation = {Yes}

Variability explicit representation = {No}

Context representation = {Development situation}

Abstraction Level = {Conceptual}

Development View

Construction process nature = {Activity-oriented}

Construction technique = {Assembly}

Tool/ Implementation = {Product storage and manipulation, Process enactment, Retrieval}

This approach is described in details in [Cossentino *et al.*, 2005] [Cossentino *et al.*, 2006] [Cossentino *et al.*, 2007] [Seidita *et al.*, 2004].

1.4.8. Comparative Analysis within the Four Views Framework

These seven approaches are compared according to the suggested framework. Table II.1.1 sums up the results of this comparison.

1.4.8.1. Usage view

Covered way. All SME approaches satisfy two of the four requirements of Seligman's method definition as they are all defined for a situational paradigm (way if thinking) and offer different meta-models to describe their approaches (way of modeling). Two approaches (the method chunks and method extension ones) specify processes that allows to organize method components within the method base. They use the MAP model in order to show precedence relationships between components. Almost all approaches (except the extension-based one) suggest some tools for supporting them. These tools are detailed in Section 1.4.8.4.

Target issues. All approaches use context-awareness to manage their components, either in the characterization of the component situation of reuse or in the definition of the project characteristics. On the same way, they all cover the reusability requirement, as it is intrinsic to the component concept. Five approaches (the method chunk, method configuration, method service, method extension, and FIPA approaches) take into account actors' goals in IS development. Thus, they respond to the intentionality issue. None of SME approaches takes explicitly into account the variability and conflict resolution issues.

Table II.1.1. SME Approaches' Review according to the Framework.

View	Attribute	Value	Method fragment approach	Method chunk approach	Method configuration approach	OPEN approach	Method service approach	Method extension approach	FIPA approach
Usage	Covered way	{Way of thinking, Way of modeling, Way of organising, Way of supporting}	{Way of thinking, Way of modeling, Way of organising}	{Way of thinking, Way of modeling, Way of organising}	{Way of thinking, Way of modeling, Way of organising, Way of supporting}	{Way of thinking, Way of modeling, Way of supporting}	{Way of thinking, Way of modeling, Way of supporting}	{Way of thinking, Way of modeling, Way of organising}	{Way of thinking, Way of modeling, Way of supporting}
	Target issues	{Variability, Intentionality, Context-Awareness, Reusability, Conflict Resolution}	{Context-Awareness, Reusability}	{Intentionality, Context-Awareness, Reusability}	{Intentionality, Context-Awareness, Reusability}	{Context-Awareness, Reusability}	{Intentionality, Context-Awareness, Reusability}	{Intentionality, Context-Awareness, Reusability}	{Intentionality, Context-Awareness, Reusability}
Subject	Knowledge construction	{Ad-hoc, Formalised}	{Ah-hoc}	{Formalised}	{Formalised}	{Formalised}	{Formalised}	{Formalised}	{Ad-hoc}
	Knowledge organisation	{Repository, Organisational process}	{Repository}	{Repository}	{Repository, Organisational process}	{Repository}	{Repository}	{Repository, Organisational process}	{Repository}
	Construction flexibility	{Selection and adaptation of a method, Modular construction of a method, Method selection in a multi-method}	{Modular construction of a method}	{Modular construction of a method}	{Selection and adaptation of a method}	{Modular construction of a method}	{Modular construction of a method}	{Selection and adaptation of a method}	{Modular construction of a method}
System	Knowledge genericity	{yes, no}	{yes}	{yes}	{yes}	{yes}	{yes}	{yes}	{yes}
	Knowledge representation	{Fragment, Chunk, Component, OPF Fragment, Service, Pattern}	{Fragment}	{Chunck}	{Component}	{OPF fragment}	{Service}	{Pattern}	{Fragment}
	Actor representation	{Yes, No}	{no}	{no}	{yes}	{yes}	{no}	{no}	{yes}
	Variability explicit representation	{yes, no}	{no}	{no}	{no}	{no}	{no}	{no}	{no}
	Context representation	{Reuse frame, Interface, Contingency factor, Development situation}	{Contingency factor}	{Reuse frame, Interface}	{Development situation}	N/A	{Interface}	{Interface}	{Development situation}
	Abstraction Level	{Conceptual, Technical}	{Conceptual}	{Conceptual}	{Conceptual}	{Technical}	{Technical}	{Conceptual}	{Conceptual}
Development	Construction process nature	{Activity-oriented, Product-oriented, Decision-oriented}	{Activity-oriented}	{Decision-oriented}	{Activity-oriented}	{Activity-oriented}	{Decision-oriented}	{Decision-oriented}	{Activity-oriented}
	Construction technique	{Assembly, Instantiation, Extension, Reduction, Agile Construction}	{Assembly}	{Assembly, Extension}	{Extension}	{Assembly, Instantiation, Agile construction}	{Assembly}	{Instantiation, Extension}	{Assembly}
	Tool/ Implementation	{Product storage and manipulation, Process enactment Retrieval, Construction support}	{Product storage and manipulation, Retrieval, Construction support}	{Product storage and manipulation, Retrieval, construction support}	{Product storage and manipulation, Process enactment, Retrieval}	{Product storage and manipulation, Retrieval}	{Product storage and manipulation, Retrieval, Construction support}	{Not exists}	{Product storage and manipulation, Process enactment, Retrieval}

The objective view is summarized in Table II.1.2.

Table II.1.2 Summary of the Usage View.

Attribute	Value	Method fragment approach	Method chunk approach	Method configuration approach	OPF approach	Method service approach	Method extension approach	FIPA approach
Covered way	Way of thinking	X	X	X	X	X	X	X
	Way of modeling	X	X	X	X	X	X	X
	Way of organizing			X			X	
	Way of supporting	X	X	X	X	X		X
Target issues	Variability							
	Intentionality		X	X		X	X	X
	Context-awareness	X	X	X	X	X	X	X
	Reusability	X	X	X	X	X	X	X
	Conflict resolution							

1.4.8.2. Subject view

Knowledge construction. Two approaches (method fragment and FIPA) emphasize the ad-hoc construction of methods. The other ones suggest formalized steps for creating new methods from method components.

Knowledge organisation. All approaches propose a method base (repository) to store the components. However, most of them do not precise how this base is structured or how a specific component may be retrieved. Two approaches propose an organisational process to manage these components in the repository, namely the method configuration and the method extension approaches.

Construction flexibility. Most of the approaches (method fragment, method chunk, OPF, method service, and FIPA are situated on the highest level of flexibility, which means they define the situational method using a modular construction. The method configuration approach also proposes to adapt a selected method with component reuse. The method extension proposes only this construction process as it is based on a pre-selected method. None of approaches suggests selecting a method path from a multi-method.

The summary of the subject view is presented in Table 11.1.3.

Table II.1.3 Summary of the Subject View.

Attribute	Value	Method fragment approach	Method chunk approach	Method configuration approach	OPF approach	Method service approach	Method extension approach	FIPA approach
Knowledge construction	Ad-hoc	X						X
	Formalized		X	X	X	X	X	
Knowledge organization	Repository	X	X	X	X	X	X	X
	Organizational process			X			X	
Construction flexibility	Selection and adaptation of a method			X			X	
	Modular construction of a method	X	X		X	X		X
	Method selection in a multi-method							

1.4.8.3. System view

Knowledge genericity. All approaches are strongly dependent of their component structure.

Knowledge representation. This attribute identifies the component type used in the SME approaches, which means Fragment (method fragment and FIPA approaches), Chunk (method chunk approach), Component (method configuration approach), OPF fragment (OPF approach), Method service (method service approach) and pattern (method extension approach).

Actor representation. Actors are sometimes taken into account as a specific concept of the approach. For instance, the method configuration approach uses the role concept to carry out the prescribed actions and modify the artefacts. The FIPA approach also investigates the role of actors in IS engineering. The OPF approach uses the term of producer to describe a core abstract method component that models someone or something that performs work units (produces work products or provides services). In this definition, a role is a specific kind of producer.

Variability explicit representation. Even if variability is intrinsic to the SME domain, this is a concept that is still not really formalized in the existing approaches. None of them define the components, or part of the component, as common or optional following the situation. Nevertheless, they do not suggest an explicit way to represent variability.

Context representation. We stated above that all approaches have a common targeted issue, which is the context-awareness. The context may be represented on several manners and some approaches use several techniques to represent this context. For instance, the method chunk approach uses reuse frames and interfaces whereas others use only the development situation (as in the method configuration and FIPA). The method fragment approach uses situation factors based on contingency

factors. Both method service and method extension approaches include the notion of interface in order to specify the context of the method components usage.

Abstraction level. Most of the approaches (method fragment, method chunk, method configuration, method extension, and FIPA approaches) are essentially conceptual. They usually propose a new way to think about method construction but without any tool support. On the contrary, others offer a more technical way to envision the problem, like the OPF and the method service approaches, which also propose a development view of the process.

Table II.1.4 gives an overview of the subject view.

Table II.1.4. Summary of the System View.

Attribute	Value	Method fragment approach	Method chunk approach	Method configuration approach	OPF approach	Method service approach	Method extension approach	FIPA approach
Knowledge genericity	Yes	X	X	X	X	X	X	X
	No							
Knowledge representation	Fragment	X						X
	Chunk		X					
	Component			X				
	OPF Fragment				X			
	Service					X		
	Pattern						X	
Actor representation	Yes			X	X			X
	No	X	X			X	X	
Variability explicit representation	Yes							
	No	X	X	X	X	X	X	X
Context representation	Reuse frame		X					
	Interface		X			X	X	
	Contingency factor	X						
	Development situation			X				X
Abstraction Level	Conceptual	X	X	X			X	X
	Technical	X			X	X		

1.4.8.4. Development view

Process model. The construction processes are essentially of two kinds: either they are activity-oriented (the method fragment, method configuration, OPF, and FIPA approaches) or decision-oriented (the method chunk, method service, and method extension approaches).

Construction technique. There are several techniques used in the approaches but the most widely used is the assembly one (which means that components are put together to produce a whole method part). Five approaches from our panel (except the method configuration and method

extension ones) use this technique. In the OPF approach, a new method is constructed by dynamic instantiation of fragments during the project. Hence, the OPF approach suggests an agile construction of methods. The method configuration and method extension approaches propose templates or patterns to instantiate in order to obtain the required component to reuse.

Tool/Implementation. Some approaches propose tools to handle their corresponding SME processes. The way of supporting is presented in the following approaches:

- The Decamerone tool supporting the method fragment approach [Harmsen *et al.*, 1994] contains the product part elements of the component;
- The SESAME tool is described in [Mirbel, 2008] and is based on the method chunk approach;
- The MC Sandbox tool belonging to the method configuration approach [Karlsson *et al.*, 2004b];
- The OPF fragment approach uses the ORF Repository tool to store the product part of the method [Firesmith *et al.*, 2001];
- The method service approach uses execution graphs and resource descriptions for implementing the resource part implemented in SO2MX tool [Guzélian *et al.*, 2007];
- The FIPA approach offers a tool for storing method fragments (a repository) [Seidita *et al.*, 2004].
- The MENTOR tool offers a way to guide the engineer through the repository of the method chunks approach [Si-Said *et al.*, 1997].

However, there are no approaches that offer a tool to support the complete method construction process. Table II.1.5 presents the development view.

Table II.1.5 Summary of the Development View.

Attribute	Value	Method fragment approach	Method chunk approach	Method configuration approach	OPF approach	Method service approach	Method extension approach	FIPA approach
Process model	Activity-oriented	X		X	X			X
	Product-oriented							
	Decision-oriented		X			X	X	
Construction technique	Assembly	X	X		X	X		X
	Instantiation				X		X	
	Extension		X	X			X	
	Reduction							
	Agile Construction				X			
	Context matching		X		X	X	X	
Tool/ Implementation	Product storage and manipulation	X	X	X	X	X		X
	Process operating			X				X
	Retrieval	X	X	X	X	X		X
	Construction	X	X			X		

1.5. Conclusion

The comparative analysis of seven SME approaches was realized in this thesis for two reasons: firstly, for identifying existing elements that can be useful in our proposal and, secondly, for establishing a possible improvements of DM methods.

The framework helps highlighting some drawbacks of the studied SME approaches.

- **Variability.** The variability concept is not explicitly taken into account in any approach; so there is no explicit representation of the common or variables parts in the current approaches.
- **Contextualisation.** Even if all SME approaches deal with the context notion, they do not propose a context model. In addition, none of the existing SME approaches suggests a methodology for defining the context of methods.
- **Actors' conflicts resolution.** Even if actors and their roles are considered by several SME approaches, they do not say how to deal with contradictory interests of stakeholders and engineers involved in IS engineering processes.
- **Approaches interoperability.** Despite some standardisation efforts of the ME community, all approaches are strongly coupled with their own notion of method component so the techniques developed in one approach are not usable in another one.
- **Component retrieval.** The case of selecting or constructing a method within multiple methods repository is not addressed.

The proposal put forward in this thesis uses the principles of SME (intentionality, context-awareness, reusability, and modularity), as well as some of the main points of SME approaches (for instance a good actor/role representation, a repository to store the components and the associated organizational process, tool supports at different levels, and so on) to define an approach targeted to the DM domain. This approach will also provide different context representations and context usages for elaborating a contextualization methodology, which will help to select & assemble the right components. The construction of situational methods will use a decision-oriented process to offer a better guidance and will allow method construction as well as method configuration. The following chapters explain the notion of method family, as well as the contextualization and configuration processes.

Chapter 2: DM Method Family

2.1. Introduction

In this Chapter, we introduce the concepts of method component and method family. A method component is a building block that represents a part of a method and could be applied and reused separately. The method family concept aims at responding to the question of how to represent and to organize method components. Current approaches of method engineering are based on the assumption that a method may be composed at the time of a specific project, composed "on the fly". The concept of method families addresses the composition in a different way: method components from the same field and having the same main usage are organized in a method family before concrete projects. That allows reducing the complexity in managing the vast variety of methods and method components. The composition of a method family exhibits the commonalities and the variability.

Based on these properties, a method family is customized in order to be applied in a given situation. A new method is obtained from the method family. We call it an *application method* which could be also called a method line.

In order to represent the method components and method families, we adapt the COMET meta-model and use the Map modeling formalism. The COMET meta-model (Section 2.2.) is based on the notion of method chunk and is used for representing the structure of method components in a modular way [Deneckère *et al.*, 2008]. We adapt this model to the case of DM methods. The Map formalism [Rolland *et al.*, 1999] is employed for the representation of the DM method family and for the organization of method components within a family. Map is an intention-oriented modeling formalism. This formalism allows specifying models in a flexible way by focusing on intentions, and on the various ways to achieve each of these intentions. This formalism is detailed in Appendix A.

A method family is a composition of method components coming from different methods of the same field. However, belonging to the same scientific field does not represent a sufficient criterion to unify several methods into a family, as such field contains multiple methods of different nature.

We postulate that different methods may be federated into a family only if they are characterized by a property that we call “teleological unity”. We introduce this notion in order to clarify the possible characteristics allowing to consider a group of methods as a family.

Teleological unity is defined by the similarity of different methods according to their usage (expressed through associated intentions). This implies that only methods having the same usage and created for the same goals could be integrated into a method family. At the same time, the differences between methods relate essentially to various ways of attaining the given goals. In that way, teleological unity is deduced based on the identification of main goals (or main intentions) of methods to be grouped into a method family. The main goal corresponds to the goal of the higher level of abstraction (See Appendix A for details of the intention description). The similarity of these goals (inspired from [Ralyté, 2001]) allows identifying the teleological proximity of methods.

This Chapter contains:

- the description of the DM method components using the proposed meta-model;
- the explanation of how the MADISE DM method family was constructed;
- the presentation of the MADISE DM method family;
- the illustration of two DM method lines obtained from the MADISE DM method family.

2.2. DM Method Components and Family Modeling

This section details the component meta-model. This meta-model is based on the COMET meta-model created for modular method modeling. Following this approach, a method is viewed as a set of modules called *method components*. This supports the adaptation and extension of existing methods and the construction of new ones depending on the specificities of the current project, which is the purpose of SME.

This section includes the meta-model of modular methods, different elements of method components, and their levels of granularity.

2.2.1. Meta-Model of Modular Methods

The goal of the proposed meta-model is to represent any method as a set of *method components* (or simply *components*). A method component is an autonomous method part and a coherent method to support the achievement of a specific activity in the process of system development.

Figure III.2.1. provides a detailed representation of our component meta-model using UML notation [UML, 2011]. According to this meta-model, a method family is seen as a set of components from different levels of granularity, the method family is itself a component of the highest level.

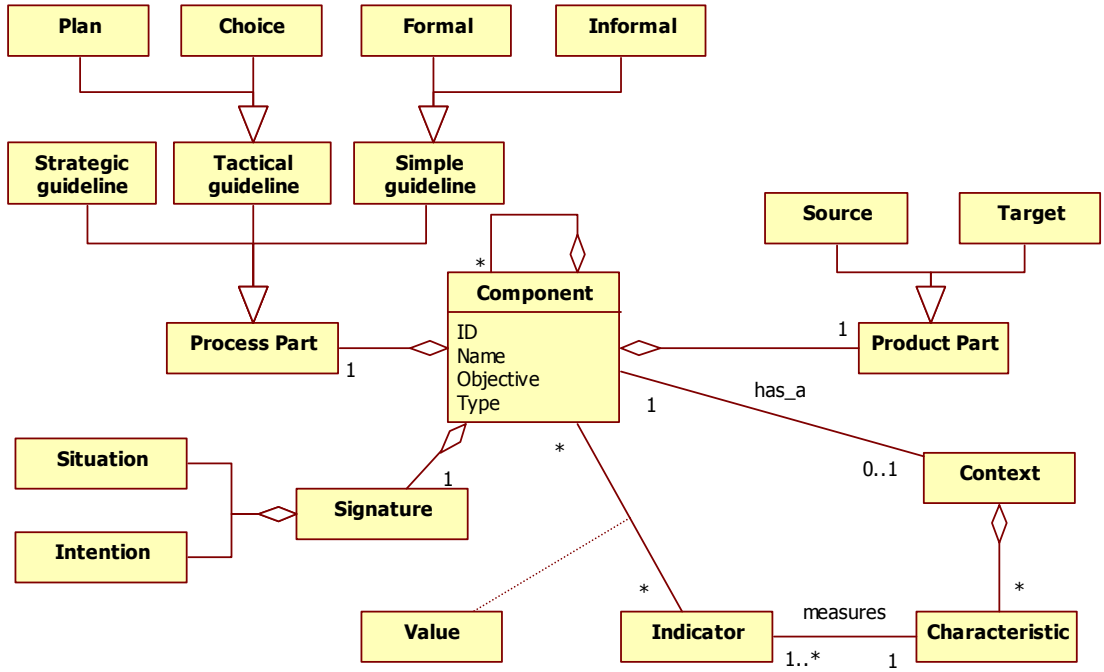


Figure III.2.1. Meta-Model of Modular Methods.

As shown in this figure, components may be an aggregation of components. Each component includes a *process part* coupled to the *product part* used by this process. For example, the component method providing a part of the process for the construction of a use case model [Jacobson *et al.*, 1992] must also provide definitions on the concepts of actors, use case, scenario, extension relationships, etc. The product part of the component contains the various items of product belonging to the product model of the method, necessary for the implementation of the approach captured in the process of the component. The component process part is represented as a directive. This is the most important part of a component because it defines an approach that the IS engineer has to follow to achieve the component objective. Another important concept in this meta-model is the *signature*. The signature represents the operating conditions of the guideline. The situation characterizes the entry point in the process of the component while the intention defines the engineering purpose the component helps to achieve. A component is also composed of a context part which allows to define some characteristics representing the situational context to reuse the component.

The following sub-sections represent the main elements of the given meta-model: method component, its signature, product part, process part, and context.

2.2.2. Method Component

Method components are obtained by dividing methods into atomic building blocks. A component is a coherent part of a method. It has several attributes: ID, name, objective, type.

- The *id* of the component allows to identify it in the component repository.
- Its *name* is a string that contains the component label.
- The *objective* attribute expresses the component purpose, which allows selecting or rejecting the component without looking at its contents.
- The *type* attribute determines the component type that can be atomic or aggregate (See Section 2.2.7.). This allows to refine the retrieved components in the repository. For the aggregate type, analyzing the component structure and choosing one of its sub-components can refine the search. For the atomic type, it cannot be sub-divided. However, the aggregate components including the given atomic component could be retrieved.

The origin method could be associated to each component. The initial method is characterized by its name, author, and reference. The method component and its relation to the initial methods are shown at Figure III.2.2.

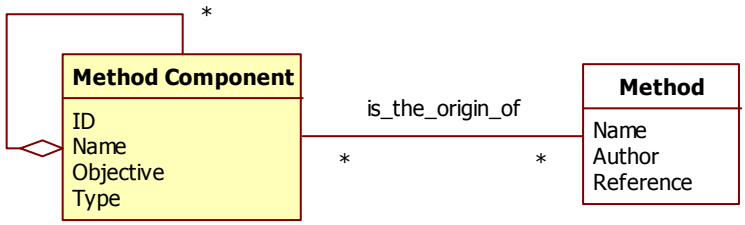


Figure III.2.2. Representation of the Method Component.

As we can see, each method component is related to at least one method. In addition, the same DM component could be initially included in more than one DM method. This is the case, for instance, of the weighting method component which is often included in more complex DM method (for instance, AHP). This component is shown at Figure III.2.3.

ID:	M.cc3
Name:	Define criteria weighting
Objective:	Provide an approach for defining relative importance of criteria.
Type:	Aggregate

Figure III.2.3. Method Component 'Define Criteria Weighting'.

The DM component M.cc3 'Define criteria weighting' allows providing an approach for defining relative importance of criteria in a given DM situation. This is an aggregate component as it is composed of other components, each of them describing a way to define weights.

2.2.3. Signature

As mentioned earlier, each component is described by a *signature*. The *signature* characterizes: (i) the conditions under which the component can be applied and (ii) the result it achieves. The component signature plays an essential role for its identification and retrieval in the repository as it defines the component goal and the conditions of its application.

A signature is defined by a pair *<situation, intention>* characterizing the application context of the component. Figure III.2.4. shows the part of the meta-model representing the signature structure.

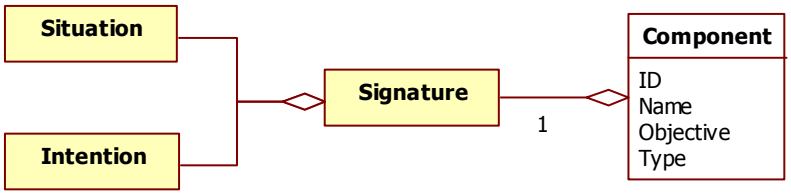


Figure III.2.4. Structure of the Method Component Signature.

DM Situation. The situation characterizes the conditions mandatory for the application of a component. In the case of the DM methods, the situation relates to the definition of different objects, used in the DM process at its different stages. For instance, it can be "alternatives are defined".

The situation is based on a product part or parts that are needed to meet the objective of the component. Thus, the situation refers to the source product part of the given method chunk. As the product part is a sub-set of concepts determined in the DM ontology, the situation includes also the elements from DMO. More precisely, it contains different classes of the DM ontology with the associate states of the corresponding objects. It can be formalized as follows:

```
Situation: <Object = «Object's state»>
```

Table III.2.1. provides the list of several objects and their states used for defining the situations of the DM method components.

Table III.2.1. Examples of the Situations' Definition for the DM Method Chunks.

Object	*Situation*
Problem	`Problem = «Defined»`
DM Object	`DM Object = «Defined»`
Alternative	`Alternative = «Validated»`

DM Intention. The *Intention* expresses a goal that the engineer wants to achieve by applying the component. The target of this goal is composed of one or more product parts composing the component. This concept describes goals that actors have within the DM process. It is represented as a taxonomy. The main intention is to find a solution of a problem having DM nature. This intention is decomposed into ones that are more detailed, for instance, define alternatives, define relative importance of criteria, and so on. Several examples of DM intentions are given in Table III.2.2.

Table III.2.2. Examples of Decision-Making Intentions.

Intention	*Description*
Define alternatives	Establish the list of alternatives available in a given DM situation.
Define criteria	Establish the list of criteria in order to allow considering different aspects for comparing alternatives.
Define relative importance of criteria	Define weights of criteria in order to take into account different relative importance of criteria.

The formulation of DM intentions is based on the model specified in Section A.1.2 of Appendix A.

An example of the method component signature is given at Figure III.2.5.

Signature <(Alternatives evaluated), Evaluate alternative by quantifying>

Figure III.2.5. Signature of the Method Component 'Define criteria weighting'.

This Figure shows the signature containing the situation in which alternatives have already been evaluated but are expressed with the qualitative indicators ('Alternatives evaluated'). The analyst apply the *by quantifying* strategy in order to obtain the quantitative values of alternatives.

2.2.4. Product Part

The product part of a component defines the product that is used for or is obtained by the execution of its process part. It is described by a product meta-model. This meta-model is a set of concepts that

can describe all the product parts of a given method. It provides a unique representation of product models belonging to the same method or to the group of similar methods.

As the *product part* is the description of input and output product models of the method component, it includes the source product part and target product part (See Figure III.2.6.).

- ***Source Product Part.*** The *source product part* defines the product situation before applying the method component. The source product part partially covers the situation included in the method component signature.

- ***Target Product Part.*** The *target product part* defines the result, which must be obtained by the method component application. The target product part is related to intentions as the last expresses the desired result in terms of objective, and the latter formalizes this result in form of product.

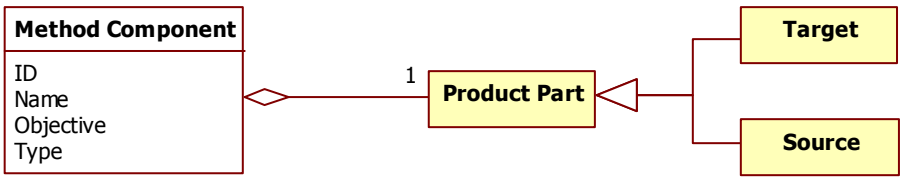


Figure III.2.6. Product Part of Method Components.

In the case of DM methods, the product part in defined using the DM ontology. The relation between the product part of the DM method components and the DM ontology is shown at Figure II.2.7.

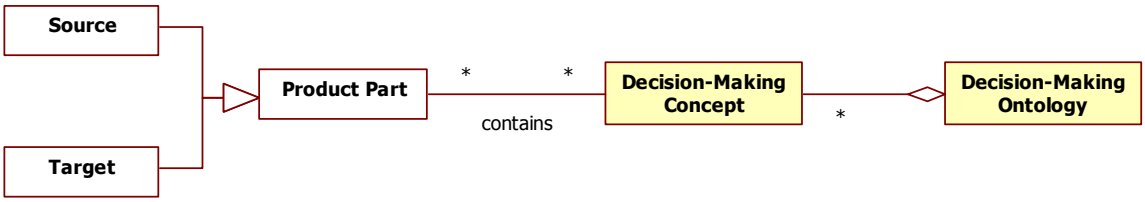


Figure II.2.7. Relation between the DM Ontology and the Product Parts of DM Method Chunks.

DM concepts correspond to different elements formalized with the DM ontology, for instance, the DM problem (choice, ranking, classification), alternative, criterion, and so on. Both source and target product parts contain different DM concepts.

An example of the product part belonging to the 'Define Criteria by Weighting' method component is given at Figure II.2.8.

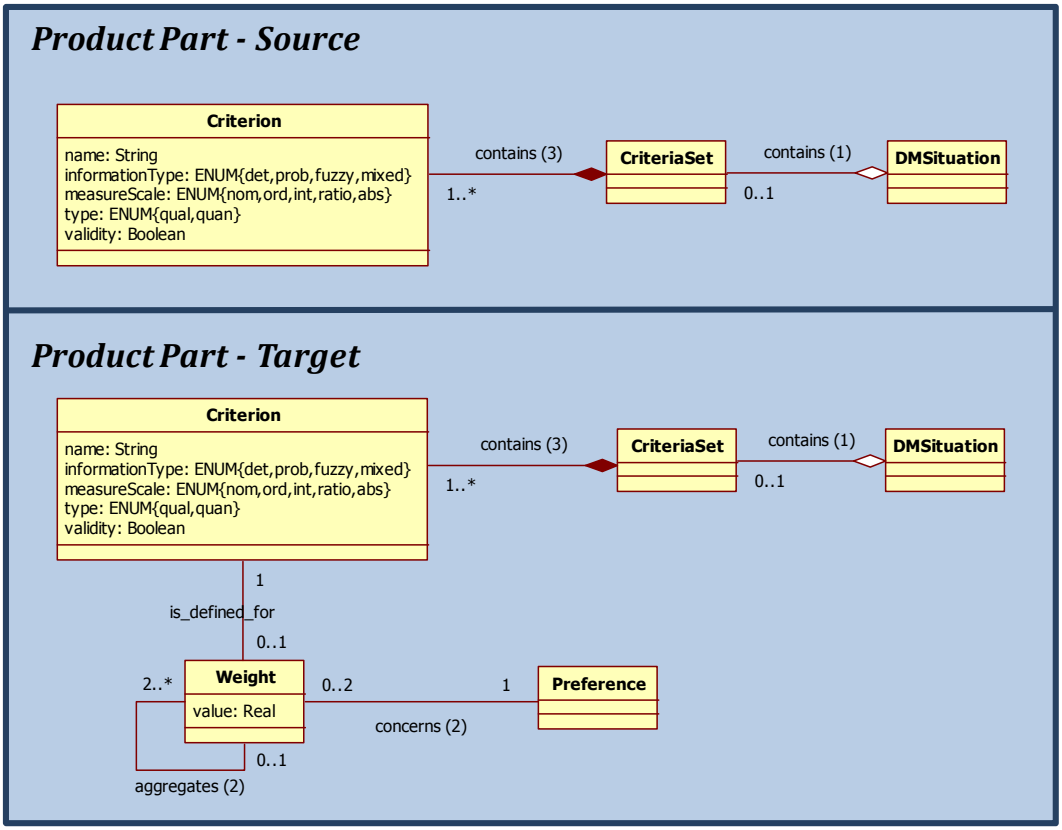


Figure II.2.8. Product Parts (Source and Target) of the Method Component 'Define Criteria Weighting'.

The criteria organized into a set are inputs (source product part) of the given component. By applying the "By weighting" strategy, criteria are analyzed in order to identify their relative importance. This leads to the transformation of the source product part (for instance, *Criterion* class) by extending it with the *Weight* class (cf. target product part).

2.2.5. Process Part

The *process part* is a description of activities executed on the input products in order to obtain the target product. It contains guidelines, which aim at explaining how to apply the method component in practice.

The process part of a component represents the process to apply to obtain the target model part. More specifically, the guideline defines the knowledge of the method to guide the IS engineer in achieving an intention in a given situation. It prescribes a process to achieve an intention. A typology of guidelines is shown at Figure III.2.9.

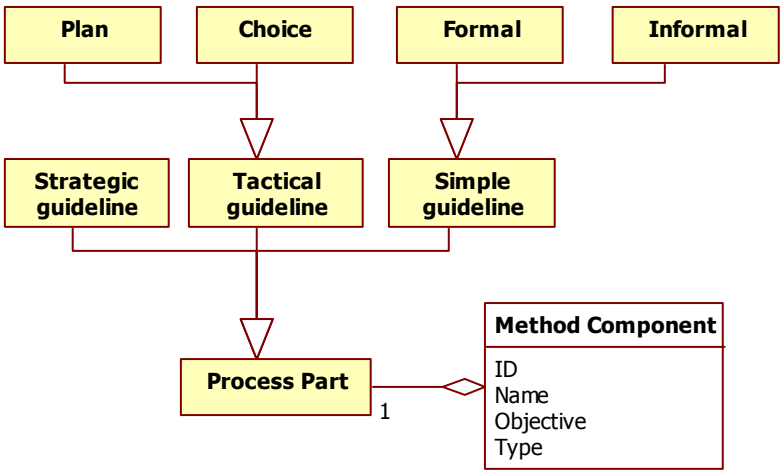


Figure III.2.9. Process Part of Method Components.

This typology classifies guidelines according to their content. Depending on the shape of their representation, their complexity and richness, they can be *simple*, *tactical* or *strategic*.

2.2.5.1. Simple guideline

A simple guideline is an atomic guideline which cannot be decomposed into sub-guidelines. It can be informal or formal. An *informal* guideline is represented as a text. A *formal* guideline is described using a formalism.

Figure III.2.10. illustrates the formal simple guideline of the 'Normalize alternative values by maximal value' method component.

Process Part

$$r_{ij} = \frac{g_j(a_i)}{\max_i \ g_j(a_i)}, i = 1, \ldots, m; j \in F$$

g_j is a criterion j
a_i is an alternative i
$g_j(a_i)$ is the evaluation of the alternative i to criterion j
r_{ij} is the normalized value of the alternative i according to criterion j
F is a set of criteria

Figure III.2.10. Simple Guideline of the 'Normalize Alternative Values by Maximal Value' Method Component.

This guideline aims at normalizing alternative values by dividing them by the maximal value according to a criterion.

2.2.5.2. Tactical guideline

A tactical guideline is a complex guideline using a tree structure to link its sub-guidelines. A guideline can be either a choice or a plan [Rolland *et al.*, 1996].

A *choice* guideline corresponds to a situation that requires the exploration of alternative options as there are situations in which an IS engineer has different ways to achieve his purpose. He must make a choice among a set of possibilities to solve the problem. Each alternative solution is described by a new guideline that meets the same intention that the choice guideline's one. These alternatives guidelines can be simple or tactical. For instance, Figure III.2.11. illustrates the choice guideline of the 'Evaluate alternatives by normalizing' method component.

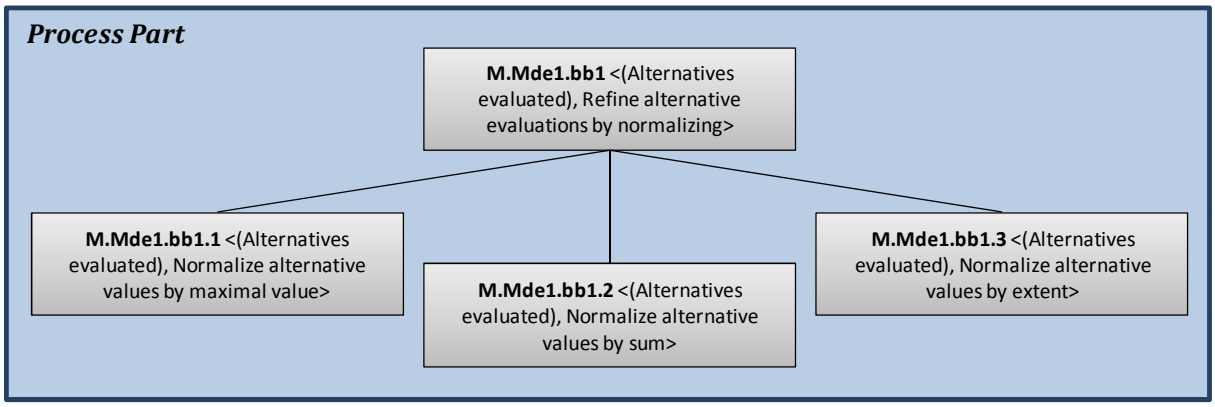


Figure III.2.11. Choice Guideline of the 'Evaluate Alternatives by Normalizing' Method Component.

The process part of the given method component contains three possible ways of normalizing (by maximal value, by sum, and by extent). Each of them represents a method component. Only one of these components is sufficient to attain the target goal which is to have normalized (that is to say comparable) values between different criteria.

A *plan* guideline corresponds to a complex problem to be solved, needing to be decomposed into a set of sub-problems. The IS engineer knows all steps that will allow him to achieve his purpose. He has a plan consisting of a set of sub-guidelines. Guidelines that compose a plan guideline can be simple or tactical.

An example of a tactical guideline plan is given at Figure II.2.12.

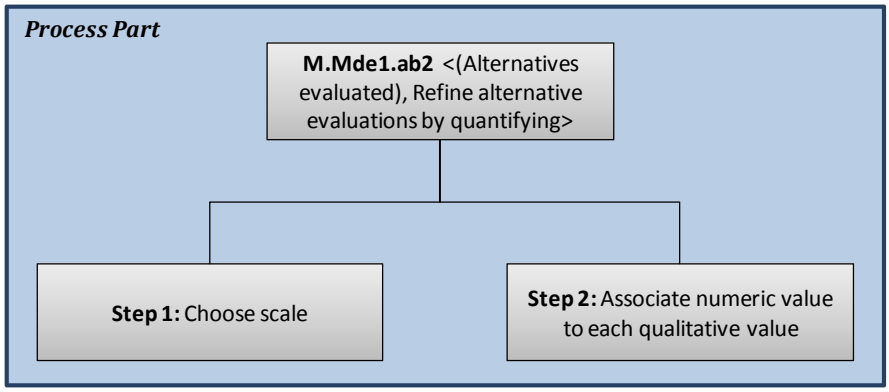


Figure III.2.12. Plan Guideline of the 'Evaluate Alternatives by Quantifying' Method Component.

This guideline indicates how to quantify alternative values expressed with the qualitative values. This leads to the execution of two steps (Choose scale and Associate numeric value to each qualitative value), each of them corresponding to a simple guideline. The analyst must carry out two steps in order to get the target product.

2.2.5.3. Strategic guideline

A *strategic* guideline, also known as a 'map' [Rolland *et al.*, 1999], allows to formalize the process model of a method by offering several possible ways to develop its products. In this case, the IS engineer has the option of choosing one approach among several suggested by the method. A strategic guideline is a complex guideline using a graph structure to link its sub-guidelines. It shows a strategic view of the product development approach based on a set of intentions and a set of strategies to meet these intentions. The strategic guideline allows expressing the multi-approaches process by offering several possible ways to satisfy his intention.

For these reasons, we have selected this model to represent method families. As mentioned above, a method family unifies several methods from the same field. Thereby, a modelling which allows to consider different possibilities to attain the same goals is needed. Thus, a method family corresponds to a strategic guideline of the highest level of abstraction modelled with a map.

Figure III.2.13. shows the MAP meta-model using the UML representation.

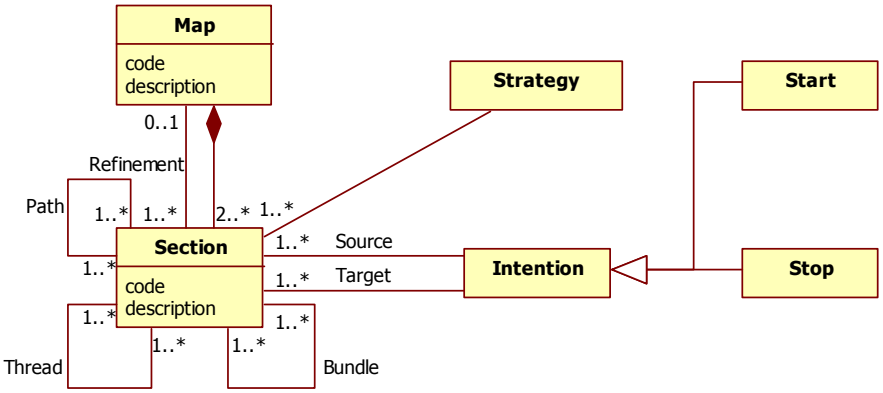


Figure III.2.13. Map Meta-Model.

The Map model enables to represent a flexible ordering of intentions. It is expressed through the combination of different intentions and strategies used for achieving these intentions. Each Map has a code and a description which allow to identify it. A map could be refined by another map if a section corresponds to another strategic guideline.

An *Intention* is a goal that can be achieved by the performance of an activity. Each map has two special intentions, *Start* and *Stop*, respectively to begin and to end the map [Rolland *et al.*, 2001]. The detailed description of the intention elements is inspired from [Rolland, 2011] [Ralyté, 2001] and detailed in Appendix A. The target intention of each section corresponds to the intention of the given method component. The source intention is related to the situation of the component.

A *Strategy* is an approach, a manner to achieve an intention [Rolland *et al.*, 2001]. Each strategy relates two intentions. The strategy concept allows, firstly, to separate the goal and the manner to achieve this goal and, secondly, to express alternative approaches for the goals achievement [Deneckère, 2001]. An instance of strategy is *By problem exploring*.

A *Section* is the basic block of the Map model. Sections belonging to the same map correspond to different method components of the given method family. A section represents a combination of two intentions and a strategy relating these intentions. In other words, a section encapsulates knowledge about a method component in a triplet <Source intention; Strategy; Target intention>, in other terms, knowledge corresponding to a particular process step to achieve an intention (the target intention) from a specific situation (the source intention) following a particular technique (the strategy). A section is characterized by a code and a description. A Section code depends on its position on the Map. The codification principles are given in Appendix A. An example can be mentioned: <*Start; By problem exploring; Identify DM Requirements*>. A section is associated to a guideline, which explains how to reach the target intention from the source one by performing the given strategy.

When a section of a map is refined by another map, it is made through the refinement relationship between the section and the map. Refinement is an abstraction mechanism by which a complex

assembly of sections at level *i+1* is viewed as a unique section at level *i.* This relationship introduces levels in the process representation as each map may be represented as a hierarchy of maps.

Sections in a map are related to each other by three kinds of relationships namely thread, path and bundle. A *thread relationship* shows the possibility for a target intention to be achieved in several ways from the same source intention. Each of these ways is expressed as a section in the map. A *path relationship* establishes a precedence relationship between sections. For a section to succeed another, its source intention must be the target intention of the preceding one. A *bundle relationship* shows the possibility for several sections having the same source and target intentions to be mutually exclusive. From this point of view, a map represents a multi-path. Method components expressed trough the sections are related by the same relationships.

MAP allows specifying process models in a flexible way by focusing on the process intentions, and on the various ways to achieve each of these intentions. Therefore, it is not imposed that once an intention is achieved the intention that immediately follows is directly undertaken. An edge enters a node if its associated strategy can be used to achieve the target intention. Since there can be multiple edges entering a node, the map is able to represent the many ways for achieving an intention.

An example of the strategic guideline is given at Figure III.2.14.

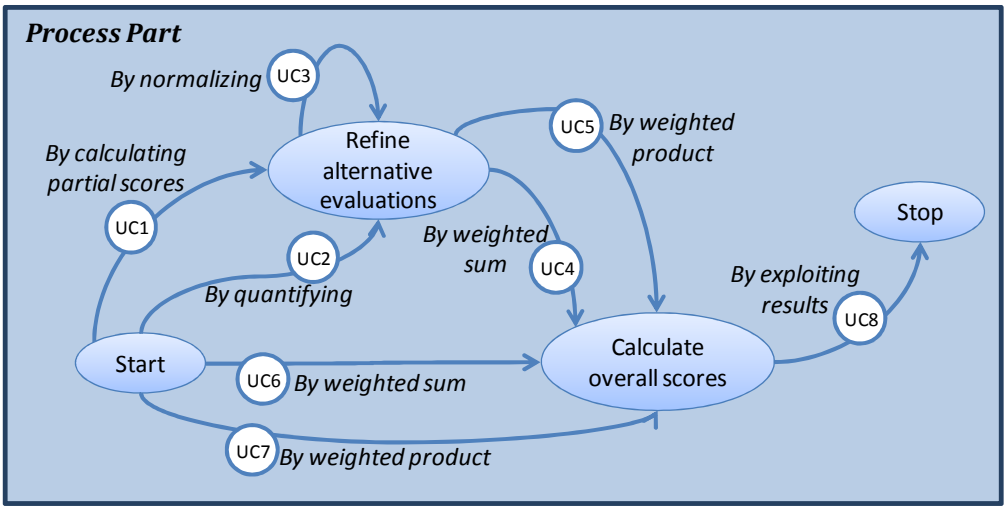


Figure III.2.14. Strategic Guideline of the 'Make Decision by Unique Criterion of Synthesis'.

This guideline is strategic as it is composed of different method components and is represented by the Map model. It shows eight components and their possible order of application. This map is a part of the DM method family proposed in this thesis and is detailed in the following.

2.2.6. Context of Method Components

Our vision is that a method component closely related to its context. The previous works on method engineering suggest neither a model of context nor a process for specifying it. We have considered these drawbacks and have suggested the notion of context formalized with the following model (See Figure III.2.15.).

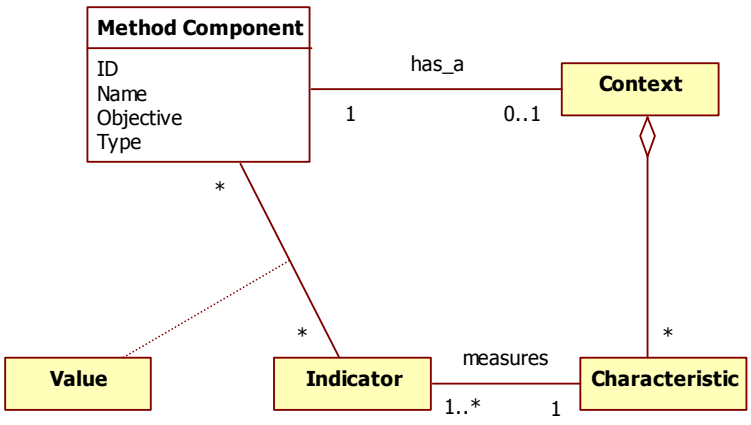


Figure III.2.15. Context Part of Method Components.

The context is composed of a set of characteristics such as cost, time, expertise degree, and so on. Each characteristic is measured by at least one indicator. A method component is evaluated according to different indicators and can have multiple values for each indicator.

By introducing context in our model, we intend to detail the reuse situation in order to define the different cases in which each method component is suitable and to provide information for searching components based on their evaluations.

The notion of context and an approach for specifying it are detailed in Chapter 3 of this Part.

2.2.7. Granularity Levels of Method Components

Method components are defined at different levels of abstraction and exist at different levels of granularity. The size of a component varies from a simple guideline to the representation of the complete method family. From this point of view, a method component can be atomic or aggregate (c. Figure III.2.16.).

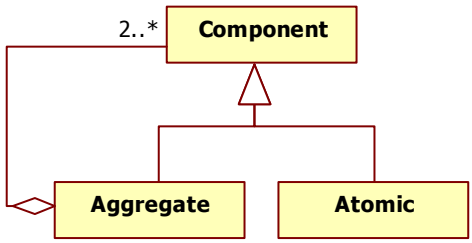


Figure III.2.16. Granularity Levels of Method Components.

An *atomic* component is a component in which the guideline is a process or an activity that cannot be divided into reusable sub-guidelines.

A component is *aggregate* only if the associated guideline is complex (tactical or strategic) and its sub-guidelines are represented by method components. The aggregate component is then composed of several sub-components. These sub-components in turn can be either atomic or aggregate. Therefore, a method family can be considered as an aggregate component of the higher level.

A *choice* tactical guideline can be an aggregate component if the alternatives composing it contain reusable guidelines and can be represented by method components. Thus, the aggregate component connects a set of sub-components through the composition link. These sub-components are represented as independent reusable method components. The execution of an aggregate component implies to select one of its sub-components, the most suitable in the given situation, and then to apply it. All these sub-components can be, in turn, atomic or aggregate.

In the same way, a method component with the *plan* tactical guideline can be atomic or aggregate. It depends on the structure of the sub-guidelines composing it. If they can be considered as reusable in another context than the original method, they can be defined as complete method components of the N-1 level of granularity. As opposed to the choice guidelines, the IS engineer must execute all sub-components of the plan guideline in order to meet his intention. However, these sub-components could be used separately in other processes.

An aggregate method component with the choice guideline is presented at Figure III.2.17.

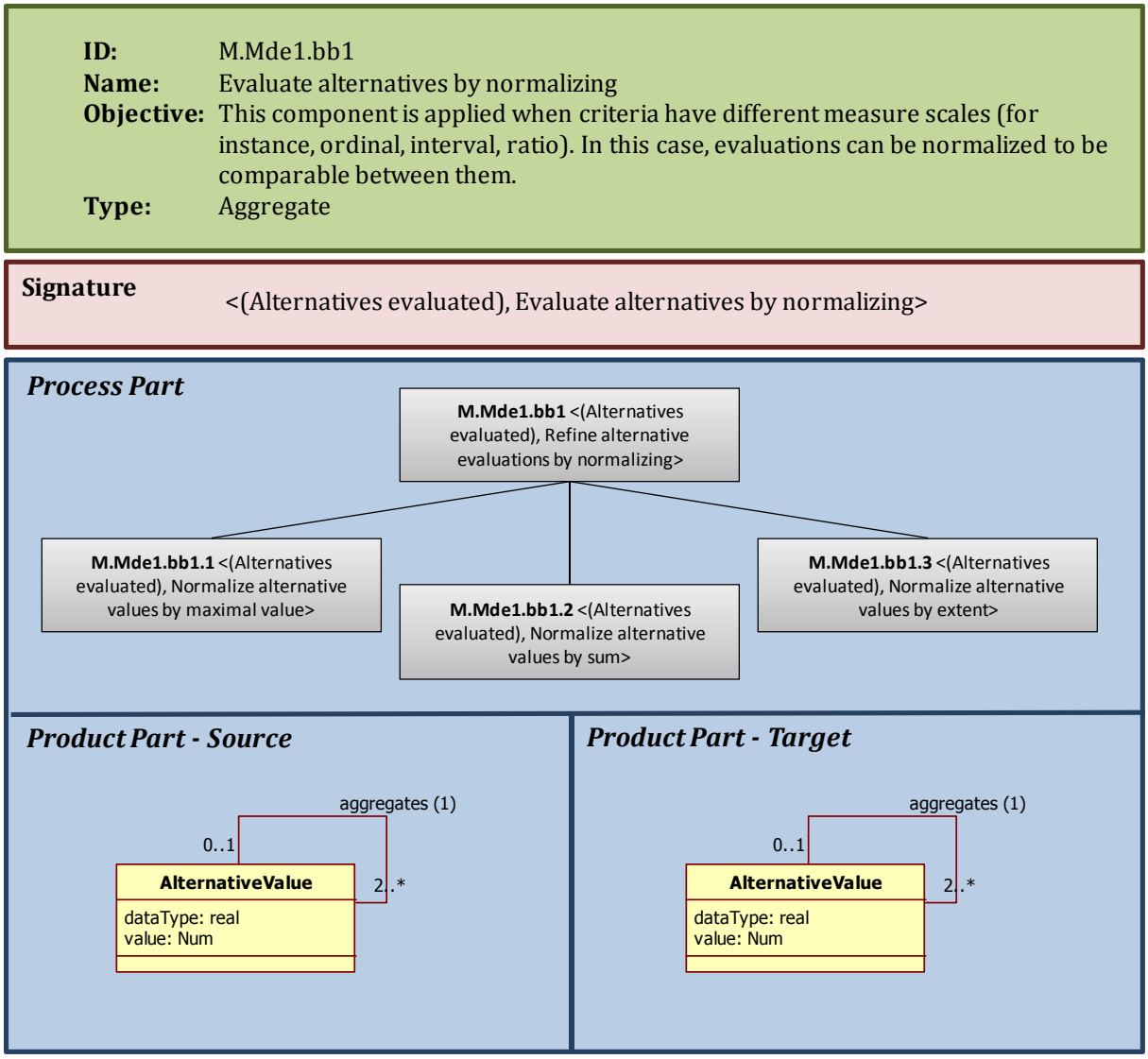


Figure III.2.17. Aggregate Method Component 'Evaluate Alternatives by Normalizing'.

The given method component is an aggregate one as it is composed of three DM method components named Normalize alternative values by maximal value, Normalize alternative values by sum, and Normalize alternative values by extent.

Figure III.2.18. shows an example of an atomic method chunk with the tactical plan guideline.

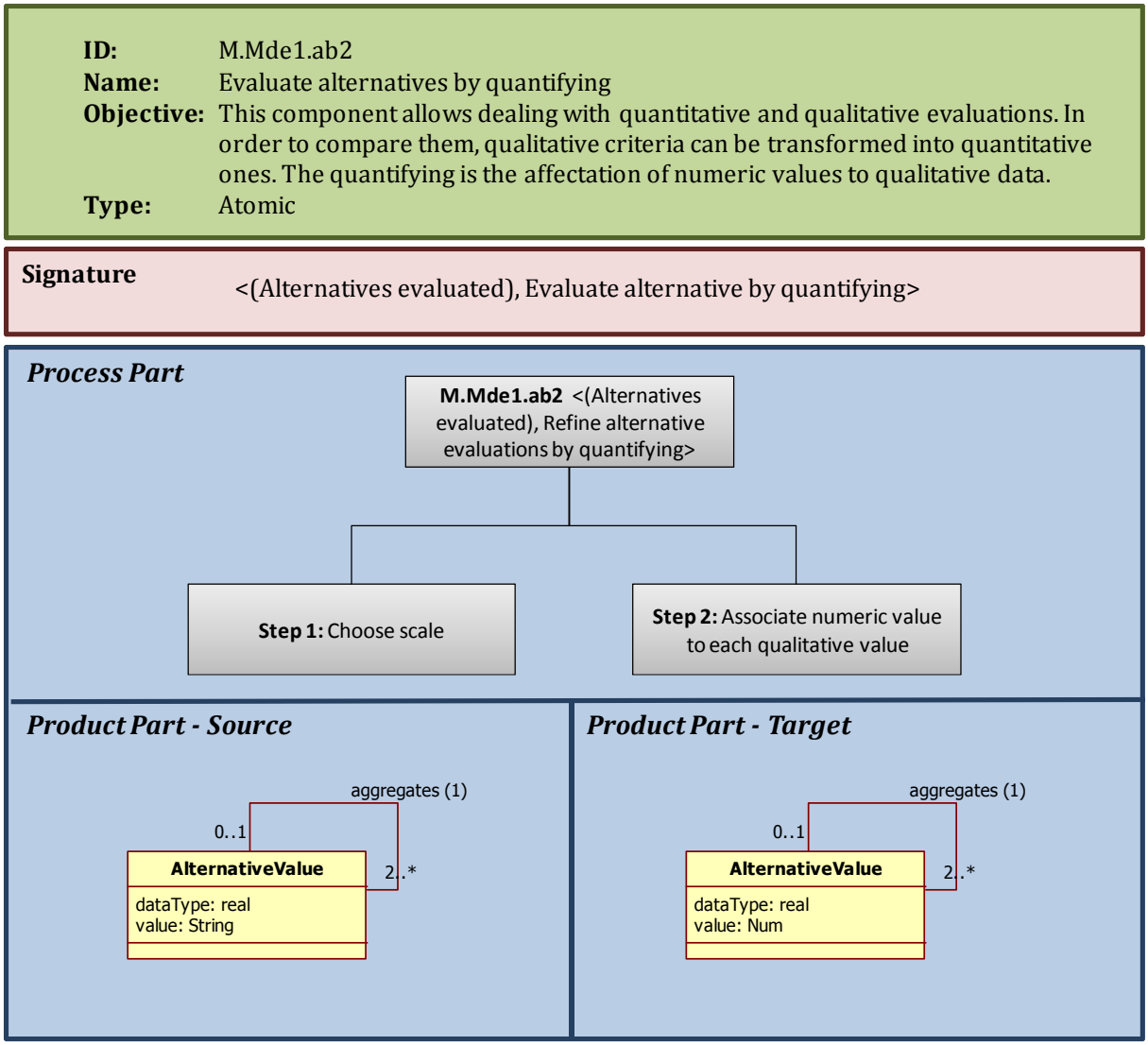


Figure III.2.18. Atomic Method Component 'Evaluate alternative by quantifying'.

This component is atomic, as it cannot be divided into simpler ones.

2.2.8. Advantages of Representing Method Families with MAP

As mentioned above, we have chosen to use the modelling formalism called MAP [Rolland *et al.*, 1999] for representing method family and for organizing method components within method families.

In our vision, a method family is considered as a method component of the highest level of abstraction. The component at this level corresponds to the strategic guideline. As mentioned above, strategic guidelines are represented with the Map model.

We chose to represent the method family with the MAP formalism to benefit from the following advantages of this representation formalism:

- **Multi-approaches dimension.** Every map path from Start to Stop represents a different approach for the method application. The concept of multi-approaches in a map is given by the multitude of paths that link Start to Stop. For example, two different paths from Start to Stop contain at least a different intention or a different strategy. Thus, the same product is constructed by two different approaches. The multi-approaches notion is rewarding because it allows more flexibility in the implementation process of the method by offering various trajectories. Moreover, this notion covers all possible approaches and does not impose ambiguous situations to IS engineers. In addition, MAP offers the possibility to combine different approaches for building the same product.
- **Intentional perspective.** Using the MAP for the representation of process models provides a clear distinction between goals to achieve and ways used to achieve them.
- **Situational vision.** The MAP formalism allows to represent the process model at different levels of granularity from the strategic level and to the operational level.
- **Different levels of abstraction.** The MAP formalism allows representing the process model at different levels of granularity from strategic to operational levels.

The choice of the MAP formalism is justified by the following reasons:

- MAP provides a simple representation of methods: indeed, it gives an overview of all aspects of a method family. In addition, representing a method family using the MAP formalism on the highest level of abstraction facilitates communication between the participants and reduces the risk of misinterpretation of components;
- MAP allows, in addition to representation, to control the process of building product models associated with the method family: the combination of guidelines with the map elements provides designers with the assistance in the product construction;
- MAP provides flexibility in the application of the process model as it does not impose any execution order: the navigation though the map is neither ordered nor sequential;
- Each path of the map from Start to Stop can give another variant of the method application;
- The notion of refinement offered by the MAP formalism provides some modularity in the implementation of the process model. Each section can be viewed as a component independent of the others;
- Guidelines associated with a map capitalize knowledge and facilitate its reuse in the construction of new methods.

Thus, using the Map model allows us to unify different method components within a unique coherent family according to the intentional paradigm.

In addition, this formalism is widely used in the SME field, for instance, in the following approaches: assembly-based method construction [Ralyté, 2001], pattern-based extension of methods [Deneckère, 2001], and paradigm-based ME method [Ralyté *et al.*, 2003].

2.3. DM Method Family Definition

In order to define DM method family, we have proceeded by the decomposition of different methods into components. The process of the DM method definition itself includes several steps organized into two stages:

- identification of DM method components and
- their further assembly into the DM method family.

These two stages are described in the following sub-sections.

2.3.1. Identification of DM Method Components

In order to identify DM method components, we have proceeded by the decomposition of different DM and other methods into components. This sub-section offers an overview of the methods used for identifying DM components and a process employed for decomposing methods. We finish by giving an illustration with the AHP method.

2.3.1.1. Spectrum of the Analyzed Methods

The input of this process includes the main groups of DM methods and also the additional activities that help in decision-making but are not directly covered by the DM methods. An example is the identification of the alternatives' list or of the criteria list. Several of these additional activities are covered by the ISE field. The spectrum of the used methods is presented at Figure III.2.19.

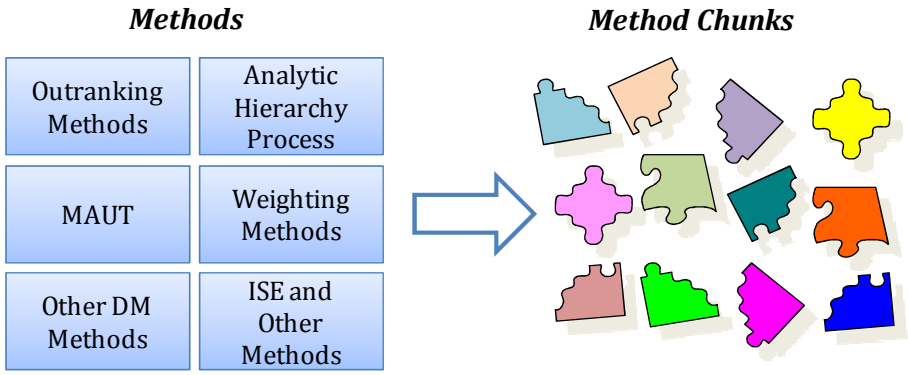


Figure III.2.19. Illustration of the DM Method Family Definition Process.

As illustrated at Figure 6.X., the initial independent methods are:

- DM methods:
  - Outranking methods,
  - Analytic Hierarchy Process (AHP),

  - Multi-Attribute Utility Theory,
  - Weighting methods;
- Other DM methods (as fuzzy methods, decision tree analysis, and so on).

We have decomposed these DM methods. Then, we have completed them with a set of several additional activities (such as alternatives discovering, criteria definition etc.) in order to cover the complete DM process. For most of them, these are the methods of analyze and synthesis (that we call ISE and other methods).

2.3.1.2. DM Method Components Identification

The DM method components were identified from the existing DM and other methods by applying the following steps: highlight elementary activities, group them with respect to several principles, and formalize method components.

The first step (*highlight elementary activities*) corresponds to the identification of DM activities. An activity describes elementary actions used for making decisions. For instance, it can be enumerate alternatives, calculate a weighted sum, calculate an aggregate value etc.

Grouping different activities into method components follows two main principles:

- ***Independency*** – Each component obtained from a method (or from a component of the highest level) must have an independent existence, that is to say it can be applied separately. This principle allows reusing components within different situations.
- ***Completeness*** – When decomposing a method into different elements (components of simple activities), the application of these elements must be sufficient for executing the initial method. This means that none activity could be omitted.

The *method component formalization* includes:

- the identification of the method component signature (the couple of situation and intention) as the most structuring elements allowing to know the preconditions of the method component execution and the goals it must reach;
- the description of the product and process parts.

In order to illustrate the method decomposition into components, we have selected the AHP method (See Section I.2.3.6. for the description of this method).

The AHP method could be seen as a set of the method components named as follows: Structure decision hierarchy, Construct the pairwise comparison matrix for goals, Construct the pairwise comparison matrix for comparing alternatives according to each criterion, Calculate alternative scores, Check consistency. These components are summed up in Table III.2.3.

Table III.2.3. DM Method Components of the AHP Method.

Name	*Signature*	*Description*
Structure decision hierarchy	<(Problem = defined), Structure decision hierarchy>	This component aims at structuring the decision hierarchy: define goals and alternatives. This is an aggregate component including: Define goals and their hierarchy, Define alternative solutions
Construct the pairwise comparison matrix for goals	<(Goals = defined), Define relative importance of goals by pairwise comparisons>	This component allows calculating the weights of goals which are considered as criteria. For this, a 1-9 scale is used and each couple of criteria (goals) is compared using this scale.
Construct the pairwise comparison matrix for comparing alternatives according to each criterion	<(Goals = defined; Alternatives = defined), Evaluate alternatives according to each goal>	By applying this component, each couple of alternatives is compared using a 1-9 scale. Thus, alternatives are evaluated according to their contribution to each goal (criterion) achievement.
Calculate alternative scores	<(Alternatives = evaluated), Define priority>	The comparison matrices are used for calculating the final score of each alternative using the defined weights.
Check consistency	<(Priority = defined), Check decision consistency>	This component helps to check decision consistency by computing a consistency measure based on the normalized comparison matrix.

2.3.2. DM Method Chunks Organization within the DM Method Family

The organization of the DM method chunks within the method family is done based on the assembly-based SME approach.

2.3.2.1. Assembly-based Approach and Assembly Operators

The assembly based approach [Ralyté *et al.*, 2001b] uses a process (assembly process model – APM) that guides the engineer in the elaboration of a requirement map and uses this map in order to select a set of related chunks. The final selection is then realized with the help of similarity measures inspired from those proposed by [Castano *et al.*, 1993] and [Bianco *et al.*, 1999]. They distinguish two types of measures: those which allow to measure the similarity of the elements of product models and those which allow to measure the closeness of process models elements.

The APM [Ralyté *et al.*, 2001b] is based on the notion of "chunk" as a representation of a method small unit. It proposes different ways to select them that match requirements as well as different strategies to assemble them. It is based on the achievement of two key intentions: *Select method chunks* and *Assemble method chunks*. Achieving the first intention leads to the selection of chunks from the method base that matches the requirements. The second intention is satisfied when the selected chunks have been assembled in a consistent manner.

The process starts by selecting candidate chunks that are expected to match the requirements expressed in a requirements map. Guidelines suggest formulating queries to the method base in order to identify the chunks that are expected to match part or the totality of the requirements. A set of strategies (*decomposition, aggregation, refinement, decomposition, aggregation)* help to refine the candidate chunk selection, but, any time a chunk has been retrieved, it can be validated by applying an *evaluation strategy*. This helps in evaluating the degree of matching of the candidate chunk to the requirements. This is based on similarity measures between the requirements map and the map of the selected chunk.

When at least two chunks have been selected, the method engineer can progress to the assembly of these chunks. Two strategies, namely the *integration strategy* and the *association strategy*, are proposed to fulfil the intention *Assemble method chunks*. The choice of the strategy depends on the presence/absence of overlaps between the chunks to assemble. Similarity measures are used to compare chunks before their assembly and to identify whether they are overlapping. This will help to choose the right strategy between the *integration strategy* and the *association strategy*.

In order to assemble DM method chunks within a method family, we have used the operators proposed in [Ralyté, 2001]:

1. *Addition* operators are used to add new elements to a product model under construction.
2. *Unification* operators are used for unifying the terminology of the models before their integration.
3. *Delete* operations are used for removing elements of the product model being constructed.
4. *Modification* operations are used for modifying product models under assembly by changing some of their elements.
5. *Merge* operators are used to merge the elements of two different product models in a new element of the model under construction.

2.3.2.2. Similarity Measures

In order to apply the assembly operators it is necessary to compare the concepts that are affected by the operator, that is to say, to measure the similarities between the elements of different product models. Like in [Ralyté, 2001], we propose to measure the similarity of concepts taking into account two parameters: their semantics and structure.

- Semantic consideration: The semantic affinity between concepts is based on real-world objects represented by concepts.
- Structural consideration: The structural affinity takes into account the structure of concepts and their relationships with other concepts.

Semantic similarity between two concepts of product models is based on the similarity of their names. This metric is called the Name affinity (NA).

The Semantics Affinity of Intentions (SAI) measures the proximity of meaning of two intentions. Since intention is composed of a verb and a target, these two parameters are involved in measuring the semantic similarity of intentions. If the verbs of the two intentions are identical or synonymous, and if the targets of these two intentions of are identical or synonymous, the semantic affinity is equal to 1, otherwise it is 0. The semantic affinity of intentions is defined as follows:

$$SAI(I_i, I_j) = \begin{cases} 1 \; if \; (I_i.\text{verb SYN } I_j.\text{verb}) \bigwedge (I_i.\text{target SYN } I_j.\text{target}) \\ 0 \; else \end{cases}$$

This formula reuses the relationship SYN of synonymy between the terms of the components defined as follows.

SYN (Synonym-of) defines the relationship of synonymy between two terms T_i and T_j, where $T_i \neq T_j$ and they which are considered synonymous. For example, we have a relationship of synonymy <Goal SYN Objective>. SYN is a symmetric relation.

When SAI (I_i, I_j) = 1, the two intentions have similar semantics which allows to merge them during the assembly of the method chunks. In the second case, the intentions have different semantics and should not be merged.

The **structural comparison** of concepts is based on the calculation of common properties and common links with other concepts. We measure structural similarity to two levels: the structural properties of concepts (or links) and concepts.

Structural similarity of Intentions (SSI) measures the percentage of intentions in two similar methods or aggregate method chunks. The calculations of the SSI are based on the research of similar intentions, that is to say, the calculation of SAI of their intentions.

$$SSI(C_i, C_j) = \frac{2 * \text{Number of Similar Intentions in } C_i \text{ and } C_j}{\sum_{i=1}^{2} \text{Number of Intentions of the Map } C_i,}$$

If SSI is greater than 0, the methods have similar intentions that can be merged during the integration. If SSI is equal to 0, the methods have no intention in common.

In the same manner, [Ralyté, 2001] defines the similarity measures for process elements of method components.

2.4. DM Method Family Description

The MADISE DM Method Family describes the generic DM process including the main activities used for DM. It is represented with the DM map.

The DM map is a collection of DM method components organized into a family in order to allow its further configuration according to a given situation. The MADISE DM Method Family represents a method component of the highest level of abstraction and corresponds to the strategic guideline. The DM method family is presented at Figure III.2.20. All components composing the DM method family are detailed in Appendix B.

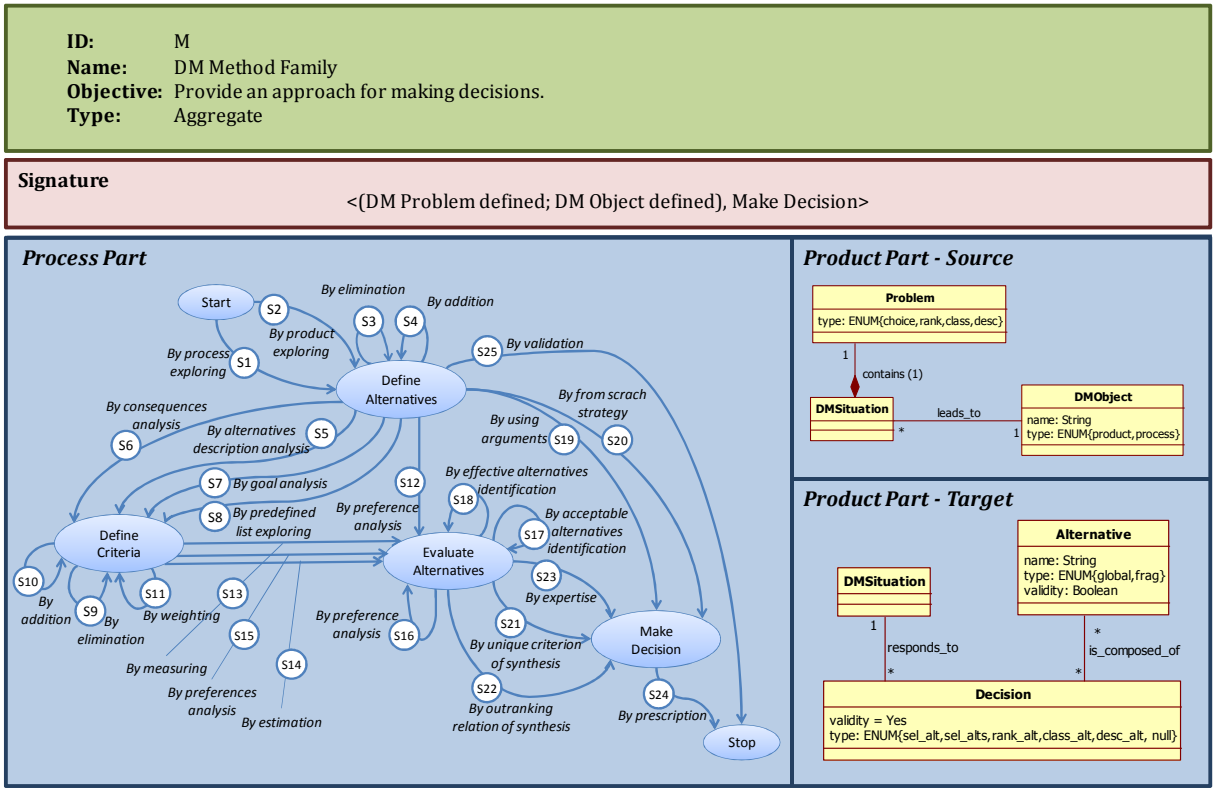


Figure III.2.20. MADISE DM Method Family (DM Map).

The DM Map contains four main intentions: *Define Alternatives*, *Define Criteria*, *Evaluate Alternatives*, and *Make Decision*. These main intentions as well as the last intention *Stop* are described in the following sub-sections.

2.4.1. Define Alternatives

The engineer starts the MADISE process by reaching the *Define Alternatives* intention. At this stage, an alternative set (or alternative family) is generated.

As commonly accepted, IS engineering processes contain two parts: a process part and a product part [Rolland, 1998]. The process part of methods captures their behavioral and procedural aspects (stages, tasks, activities, and schedules) [Agerfalk, 2003]. The product part of a method takes into account its structural and static aspects (e.g., requirements documents, models, and other deliverables) [Agerfalk, 2003]. The engineer could explore both process and product variability in

order to find possible alternatives. He starts the DM process by choosing one of the following sections S1 and S2: (i) S1 allows to *Define alternatives list By process exploring;* for instance, this section is used in intention-based approaches (based on MAP formalism) in order to select a strategy which is an action, or a process part [Rolland *et al.*, 1999]. (ii) S2 allows to *Define alternatives list By product exploring*; a usual example is the requirements prioritization, in which requirements are the product parts [Ngo-The *et al.*, 2005] [Ruhe, 2003] [Karlsson *et al.*, 1997].

Alternatives can be complementary or exclusive to each other. Once the initial alternatives list is defined, it can be refined by the two following sections: The *By elimination* strategy is applied in S3 when one or more alternatives forming the initially defined set are evaluated by the engineer as non-realistic or non-feasible [Roy, 1996] [Chakhar, 2006]. In this case, they are removed from the set and not anymore studied. An example of this section is the requirements review before applying AHP in the cost-value approach of requirements prioritization [Karlsson *et al.*, 1997]. The *By addition* strategy (section S4) is available when some alternatives could be added to the initial set by searching complementary alternatives. Such a strategy is used in [Ralyté *et al.*, 2003]: the *Refinement strategy* proposes to search for another candidate method component within the assembly-based method engineering approach.

An overview of the corresponding DM method components is given in Table III.2.4.

Table III.2.4. DM Components Used for Defining Alternatives.

Section	*Name*	*Signature*	*Type*
S1	Define alternatives list by process exploring	<(DM Object defined), Define alternatives by process exploring>	Aggregate
S2	Define alternatives list by product exploring	<(DM Object defined), Define alternatives by product exploring>	Aggregate
S3	Refine alternative list by elimination	<(Alternatives defined), Define alternatives by elimination>	Atomic
S4	Refine alternative list by addition	<(Alternatives defined), Define alternatives by addition>	Atomic

2.4.2. Define Criteria

The *Define Criteria* intention is not mandatory. The engineer selects it if he wants to arbitrate between alternatives based on multiple factors. At this stage, a set of criteria (at least one criterion) for alternatives evaluation is defined.

From the *Define Alternatives* intention, the intention *Define Criteria* can be achieved following four strategies. The *By alternatives description analysis* strategy (section S5) can be used in the case where alternatives have intrinsic characteristics, which can be considered as criteria. The engineer analyzes them in order to identify those that are important for arbitrating between alternatives. One example is the requirements characterization in [Karlsson *et al.*, 1997]. The *By consequences analysis*

strategy (section S6) deals with alternatives consequences, which are effects produced by alternatives if they are chosen. Future properties of alternatives and their effects on the decision problem are analyzed in order to identify possible criteria. The *By goal analysis* strategy (section S7) is applicable when an actor participating in the DM process has goals with regard to the decision problem. Each alternative can be measured according to its capability to contribute to these goals. In this case, goals become criteria. Goals as criteria are used in [Kornyshova *et al.*, 2007c] for business processes prioritization and in [Maiden *et al.*, 2002] for requirements prioritization. *By predefined list exploring* (section S8), engineers investigate the list of the project characteristics which are common in ISE and select the suitable ones.

Once the first set of criteria is selected, it can be refined with other strategies. The *By elimination strategy* (section S9) is used when elaborating the set of criteria, as the decision maker must comply with some rules to be coherent. For instance, in order to eliminate criteria, the IS engineer considers their set using the *SMART* method [Crowe *et al.*, 1997]. Criteria must be specific, measurable, actionable, relevant, and timely. An application of the SMART method to business processes is shown in [Crowe *et al.*, 1997]. The *By addition* strategy (section S10) is applied when criteria can complement each other. In this case, a criterion is added following the analysis of existing ones. The *By weighting* strategy (section S11) deals with weights assignment to the decision criteria. Weights are assigned when the engineer wants to define relative importance of criteria. The engineer can select one of the following techniques: (i) by simple attribution [Baudry *et al.*, 2002] [Bouyssous, 2001], (ii) by indentifying the first criterion to enhance [Mustajoki *et al.*, 2005], (iii) by trade-off technique [Keeney, 1999], (iv) by importance analysis [Keeney, 1999], or by pair-wise comparison [Saaty, 1980]. The last one is commonly used in the requirements prioritization within AHP [Karlsson *et al.*, 1997] [Maiden *et al.*, 2002].

DM method components used for criteria definition are summed up in Table III.2.5.

Table III.2.5. DM Components Used for Defining Criteria.

Section	*Name*	*Signature*	*Type*
S5	Define criteria by alternatives description analysis	<(DM Problem defined; DM Object defined; Alternatives defined), Define criteria by alternatives description analysis>	Atomic
S6	Define criteria by consequences analysis	<(DM Problem defined; DM Object defined; Alternatives defined), Define criteria by consequences analysis>	Atomic
S7	Define criteria by goal analysis	<(DM Problem defined; DM Object defined; Alternatives defined), Define criteria by goal analysis>	Atomic
S8	Define criteria by predefined list exploring	<(DM Problem defined; DM Object defined; Alternatives defined), Define criteria by predefined list exploring>	Atomic
S9	Refine criteria list by elimination	<(DM Problem defined; DM Object defined; Alternatives defined, Criteria defined), Define criteria by elimination>	Atomic
S10	Refine criteria list by addition	<(DM Problem defined; DM Object defined; Alternatives defined, Criteria defined), Define criteria by addition>	Atomic
S11	Define criteria weighting	<(Criteria defined), Define criteria by weighting>	Atomic

2.4.3. Evaluate Alternatives

The *Evaluate Alternatives* intention aims at constructing the evaluation matrix (or decision matrix) [Roy, 1996] [Chakhar, 2006]. There are several ways to evaluate alternatives:

From *Define Alternative*, alternatives can be evaluated using the *By preferences analysis* strategy (section S12). An engineer determines preferences between two alternatives a and b. [Roy, 1996] defines four elementary relations: (i) Indifference: *a***I***b* – *a* and *b* are equivalent; (ii) Strict preference: *a***P***b* – *a* is strictly preferred to *b*; (iii) Weak preference: *a***Q***b* – *a* is weakly preferred to *b*; (iv) Incomparability: *a***R***b* – *a* and *b* are not comparable. For instance, a strict preference relation is present in AHP method applied for requirements prioritization [Karlsson *et al.*, 1997] [Maiden *et al.*, 2002].

From the *Define Criteria* intention, alternatives may be evaluated following three strategies: by measuring, by estimation, and by preferences analysis.

By measuring (section S*13)*. Measuring is an activity that uses a metric definition in order to produce a value [Martín *et al.*, 2003]. A measure is a number assigned to a characteristic by making a measurement [Martín *et al.*, 2003]. A measuring method is a logical sequence of operations allowing the alternative estimation. Measuring methods are objective as the evaluation is based on numerical rules.

By estimation (section S14). Alternatives could also be evaluated using heuristics. An actor evaluates alternatives according to subjective criteria, for instance based on his opinion. The estimation is subjective as the evaluation involves human judgment [Martín *et al.*, 2003]. The evaluation method type (objective or subjective) that depends on the nature of the operations used to evaluate an alternative may be found in [ISO/IEC 15939, 2002].

By preferences analysis (section S15). An actor determines preferences between two alternatives a and b according to a criterion. For instance, the engineer can compare two requirements according to the cost criterion.

Once the evaluation matrix is constructed, alternatives evaluations can be enhanced by the three following strategies.

The *By effective alternatives identification* strategy (section S18) allows removing dominated alternatives in order to keep "effective" ones. An alternative is *dominated* if its evaluations according to all criteria are worse or at least the same that those of another alternative [Vincke, 1989] [Chakhar, 2006].

The *By acceptable alternatives identification* strategy (section S17) allows to qualify several alternatives as non-acceptable and to remove them from the alternative set. An acceptance threshold is established for a criterion. Such a technique is used in the WinWin method for requirements prioritization [Ruhe, 2003].

By using the *By preferences analysis* strategy (section S16), the engineer can enhance preferences analysis by defining complementary parameters such as preference threshold, indifference threshold, veto threshold [Roy, 1996] [Chakhar, 2006]. These parameters are used in outranking decision-making methods, for instance for business processes prioritization in [Kornyshova *et al.*, 2007c].

The corresponding components are given in Table III.2.6.

Table III.2.6. DM Components Used for Evaluating Alternatives.

Section	*Name*	*Signature*	*Type*
S12	Evaluate alternatives by preferences analysis	<(Alternatives defined), Evaluate alternatives by preferences analysis>	Atomic
S13	Evaluate alternatives by measuring	<(Alternatives defined, Criteria defined), Evaluate alternatives by measuring>	Atomic
S14	Evaluate alternatives by estimation	<(Alternatives defined, Criteria defined), Evaluate alternatives by estimation>	Atomic
S15	Evaluate alternatives by preferences analysis according to a criterion	<(Alternatives defined, Criteria defined), Evaluate alternatives by preferences analysis>	Atomic
S16	Refine alternative evaluations by preferences analysis	<(Alternatives evaluated, Criteria defined), Evaluate alternatives by preferences analysis>	Atomic
S17	Refine alternative list by acceptable alternatives identification	<(Alternatives evaluated), Refine alternative list by acceptable alternatives identification>	Atomic
S18	Refine alternative list by effective alternatives identification	<(Alternatives evaluated), Refine alternative list by effective alternatives identification>	Atomic

2.4.4. Make Decision

At this stage, a prescription for a decision is made.

By using arguments (section S19). In several approaches, decisions are based on an argument set [Grosz *et al.*, 1997] [Rolland *et al.*, 2000] [Rolland *et al.*, 1999]. The decision-maker is guided between alternatives by arguments, which indicate the alternative to select depending on the given parameters.

By "From scratch" strategy (section S20). A decision can be made by a decision-maker 'on the fly' without using a DM method. This kind of decision, allowed with the "From scratch" strategy, is specific to each DM situation and highly depends on the decision-maker skills and experience.

The method-based approach deals with the transformation of partial values (alternatives values according to different criteria) into an aggregated one. There are two main aggregation approaches using two different strategies: unique criterion of synthesis and outranking relation of synthesis.

By unique criterion of synthesis (section S21)**.** This approach consists of building a single criterion from a criteria set by using an aggregation function [Roy, 1996]. The aggregation function can be an addition or a multiplication function. The most known one is the method based on weighted sum. Weighting methods include SMART (Simple Multiattribute Technical Rating) [Keeney, 1999], SWING [Mustajoki *et al.*, 2005], and Trade-off weighting [Keeney, 1999]. Another MC method from this group is MAUT [Keeney et al., 1993], proposed by H. Raiffa and R.L. Keeney. According to MAUT, a utility function is established for each criterion. Then, the partial utility functions are aggregated to a multiattribute utility function representing either an addition, or a multiplication of the partial functions. All alternatives are evaluated by using this function. The alternative, which maximizes the utility, is selected. These methods are available for choice, ranking or classification decision problems. This component has a strategic guideline. It is shown at Figure III.2.21.

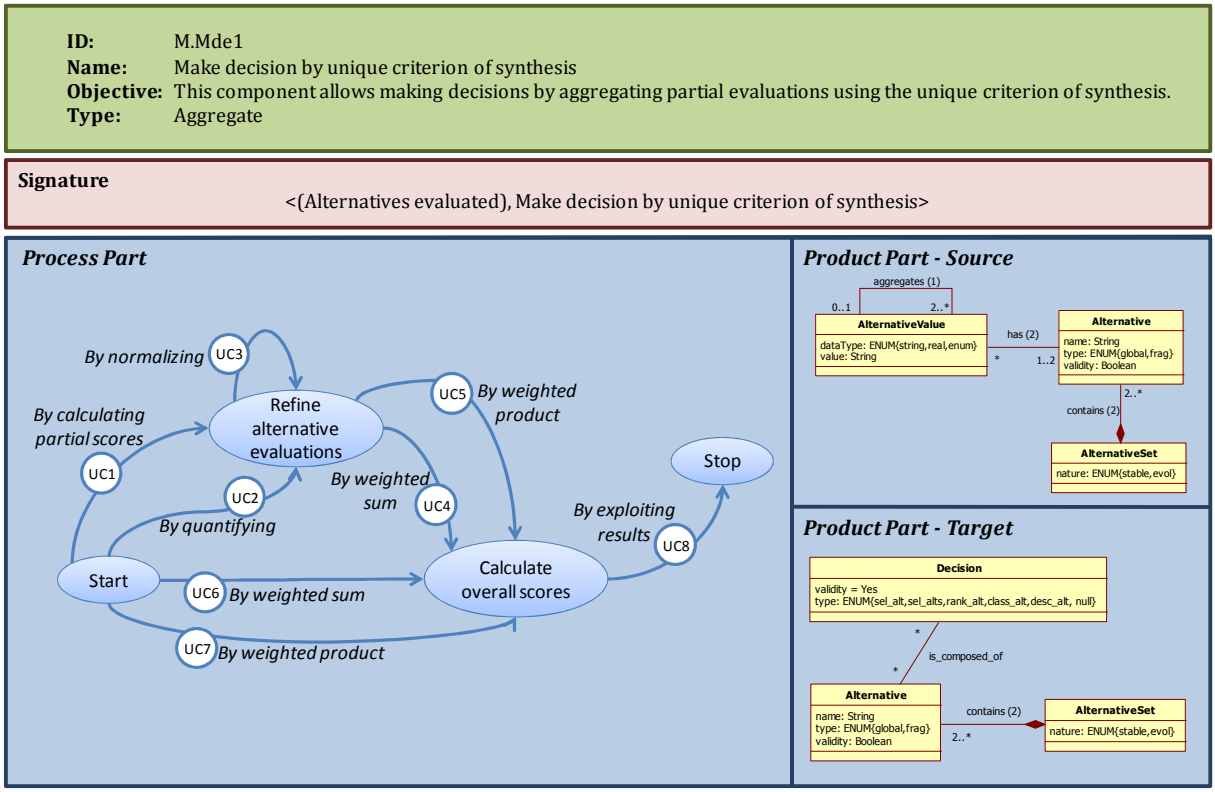


Figure III.2.21. Method Component 'Make Decision by Unique Criterion of Synthesis'.

By outranking relation of synthesis (section S22). Outranking methods [Roy, 1996] are inspired by the theory of the Social Choice. The most known outranking method is ELECTRE (Elimination And Choice Corresponding to Reality). Outranking indicates the degree of dominance of one alternative over another. Outranking methods are based on step-by-step identification of decision makers' preferences. Decision makers formulate their preferences and then a detailed analysis allows decision-making for one of the base problems (choice, ranking or classification). Outranking methods are used for method components selection [Kornyshova et al., 2007a] and for business process prioritization [Kornyshova *et al.*, 2007c]. The given component also contains a strategic guideline (See Figure III.2.22.).

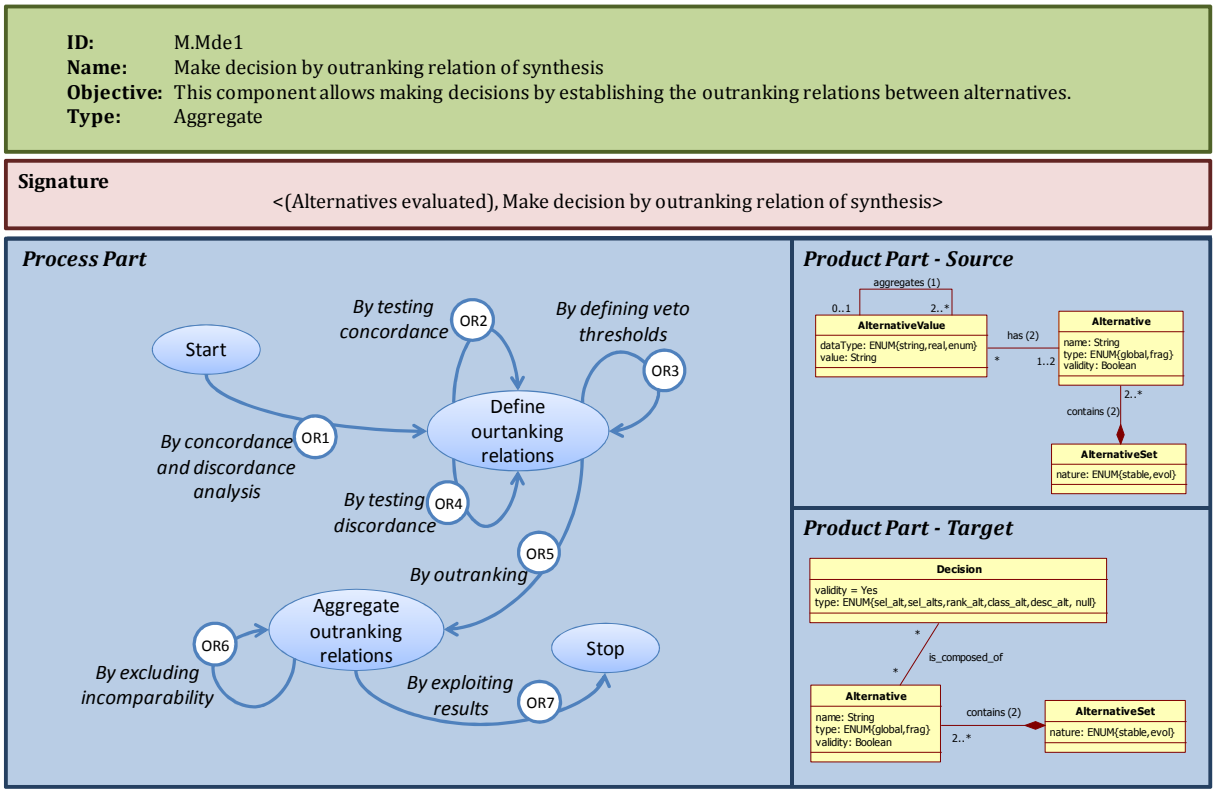


Figure III.2.22. Method Component 'Make Decision by Outranking Relation of Synthesis'.

By expertise (section S23). Experience may be sufficient to make decision, in particular if the exactly same situation has already been met. Then the *By expertise* strategy (section S24) can be used.

These components are summed up in Table III.2.7.

Table III.2.7. DM Components Used for Making Decisions.

Section	*Name*	*Signature*	*Type*
S19	Make decision by using arguments	<(Alternatives defined), Make decision by using arguments>	Atomic
S20	Make from scratch decision	<(Alternatives defined), Make decision by "From scratch" strategy>	Atomic
S21	Make decision by unique criterion of synthesis	<(Alternatives evaluated), Make decision by unique criterion of synthesis>	Aggregate
S22	Make decision by outranking relation of synthesis	<(Alternatives evaluated), Make decision by outranking relation of synthesis>	Aggregate
S23	Make decision by expertise	<(Alternatives evaluated), Make decision by expertise>	Atomic

2.4.5. Stop Decision-Making Process

This stage deals with the validation of DM results.

Once a decision is made, the engineer generates a prescription for decision based on the usage of the DM Map (*By validation*). In this case, the DM process is stopped by validation. Some metrics (for instance, a consistency index in [Karlsson *et al.*, 1997]) can be used to checking if the DM result is valid. Otherwise, the DM process continues by returning to the previous stages.

From *Define Alternative*, the DM process may be stopped when the number of identified alternatives is limited, for instance one. The engineer validates or does not the initially obtained alternative set as a decision.

Two method components are described in Table III.2.8.

Table III.2.8. DM Components Used for Stopping a DM Process.

Section	*Name*	*Signature*	*Type*
S25	Prescribe decision by validation	<(Decision made), Prescribe decision by validation>	Atomic
S26	Validate the existing alternatives	<(Alternatives defined), Prescribe decision by validation>	Atomic

2.4.6. Navigation through the DM Map

In this section, we explain the simple navigation through the DM Map.

The MAP formalism helps to use the MADISE DM family in a flexible manner. At each stage, the Map indicates all possible trajectories to go ahead and gives information about how to choose a trajectory, but does not impose any one. The engineer can select between different intentions or strategies which constitute possible trajectories. The engineer makes these choices and composes a set of Map sections that are executed consecutively. In this manner, he dynamically constructs a DM method suited to the given situation. For instance, one possible trajectory is to *Define alternatives*, *Define criteria*, *Evaluate alternatives*, *Make decision* by methods application (for instance multicriteria method AHP [Saaty, 1980]). Another one is to *Define alternatives* and then *Make decision* directly (by ad hoc or by using arguments strategies) like in [Grosz *et al.*, 1997] [Rolland *et al.*, 2000]. Therefore, the MADISE Map provides the IS engineer with a complete set of guidelines for both trajectory selection and sections execution. It can be used each time an IS engineer meets a DM situation.

An engineer using the DM Map can restart any intention. For instance, several situations could be mentioned:

- ***Identified criteria are not consistent with alternatives*** (for instance, if goals are selected as criteria and given alternatives do not much these goals). The engineer can return to the *Start* intention (to redefine the initial set of alternatives) or to the *Define Alternative* intention (to decide between defining a new set of criteria of making decisions without comparing alternatives according to any criterion). This situation is also confirmed by the proposal of [Vincke, 1989] who asserts that the alternative set definition depends on criteria definition.
- ***Evaluating alternatives is not possible*** (for instance, if the alternative's values cannot be calculated). In this case, the *Define Criteria* intention could be attained once again in order to identify a new set of criteria. The previously mentioned strategies could also be applied.
- ***Decisions are not accurate*** (for instance, if the result is a set of selected alternatives and the goal is to have only one alternative). In this case, the engineer could go back to the *Evaluate Alternative* intention and repeat the *By unique criterion of synthesis* strategy on the obtained alternative set in order to refine the solution. For example, the WinWin requirements prioritization approach uses multiple iterations in order to make results more exact [Ruhe, 2003].
- ***Decisions are not made*** (for instance, when the aggregated values are the same for all alternatives). To deal with this situation, the engineer could return to the *Start*, *Define Alternative* and *Define Criteria* intentions. For example, [Vincke, 1989] mentions that the alternative set depends on the aggregation method. Thus, the engineer can begin anew the *Define Alternative* intention if the defined alternatives are not suitable for the selected aggregation method.

The engineer can also stop the navigation at any point of the map: from the *Define Alternative* intention if there is zero or only one identified alternative or from the *Evaluate Alternatives* intention if the result of alternatives evaluation is not satisfactory.

In this section, we have explained how the DM Map can be used by navigation that is the classic way of using maps. However, this way has some drawbacks. A configuration process will be suggested in Chapter 4 in order to provide a comprehensive way to use method families in general and the DM method family in particular. However, the configuration process requires the definition of the DM components context which is the topic of the following section.

2.5. Illustration

In this section, our goal is to show that existing DM processes could be expressed through the MADISE DM method family. For doing this, each DM process must be represented as a MADISE application method (i.e. a sub-set of MADISE sections). In order to illustrate this, we have chosen two existing and well known DM processes: the cost-value approach for requirements prioritization [Karlsson *et al.*, 1997], and tool selection from the Rational Unified Process (RUP) [RUP, 2007]. Their DM processes are captured and expressed as method lines (or application methods) in the following sub-sections. Brief conclusions are given in sub-section 2.5.3.

2.5.1. Application Case: the Cost-Value Requirements Prioritization Approach

The cost-value requirements prioritization approach [Karlsson *et al.*, 1997] aims at ranking requirements using the AHP DM method. The AHP (Analytic Hierarchy Process) proposed per T.L. Saaty [Saaty, 1980] is described in Part I. As a shot reminder, this method is based on pair-wise comparison between alternatives and/or criteria and aggregation of comparison results into a quantitative indicator (score).

Figure III.2.23. shows the set of DM method components corresponding to the cost-value approach.

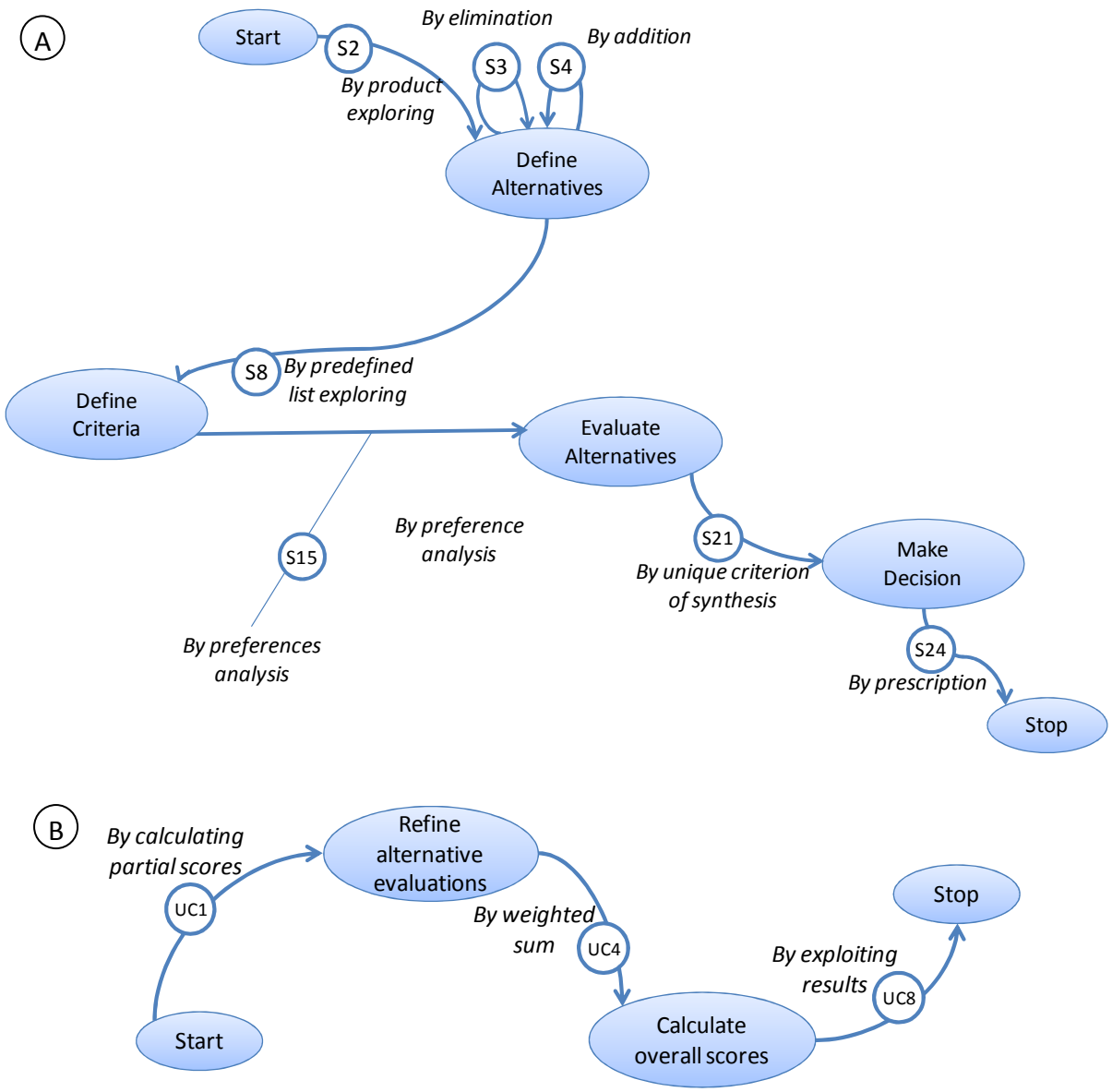


Figure III.2.23. DM Application Method of the Cost-Value Requirements Prioritization Approach: A) From the DM Method Family; B) From Method Component 'Make Decision by Unique Criterion of Synthesis'.

The cost-value approach trajectory through the MADISE Map is as follows. The product based strategy is available for identifying candidate requirements (the *By product exploring* strategy is selected). This approach suggests reviewing candidate requirements for ensuring their completeness and correctness. Therefore, requirements can be added to or removed from the initial set (The *By elimination* and *By addition* strategies are selected). The approach defines two criteria describing requirements: relative cost and relative value. These criteria are predefined by the cost-value approach (The *By predefined list exploring* strategy is selected). Actors (users and engineers) express their preferences by pair-wise comparison for defining the relative value and cost of candidate requirements (The *By preferences analysis* strategy is selected). The aggregated value obtained by AHP application is used for ranking requirement. The cost-value approach uses also a cost-value diagram in order to assist DM (The *By unique criterion of synthesis* strategy is selected). A consistency index is calculated in order to check the result validity (The *By validation* strategy is selected). The DM components used by the cost-value approach are given in Table III.2.9.

Table III.2.9. DM Method Line of the Cost-Value Approach: DM Components List.

Section	***Name***
S2	Define alternatives list by product exploring
S3	Refine alternative list by elimination
S4	Refine alternative list by addition
S8	Define criteria by predefined list exploring
S15	Evaluate alternatives by preferences analysis according to a criterion
S21	Make decision by unique criterion of synthesis
UC1	Refine alternative evaluations by calculating partial scores
UC4	Calculate overall scores by weighted sum
UC8	Make decision by exploiting results
S24	Prescribe decision

2.5.2. Application Case: the Tool Selection in RUP

The second example deals with tool selection and was taken from the RUP [RUP, 2007]. The RUP provides a wealth of guidance on software development practices. One of these practices is “Select and Acquire Tools”. This task guides the selection of tools that fit project needs.

The set of DM method components representing the tool selection in RUP is illustrated at Figure III.2.24.

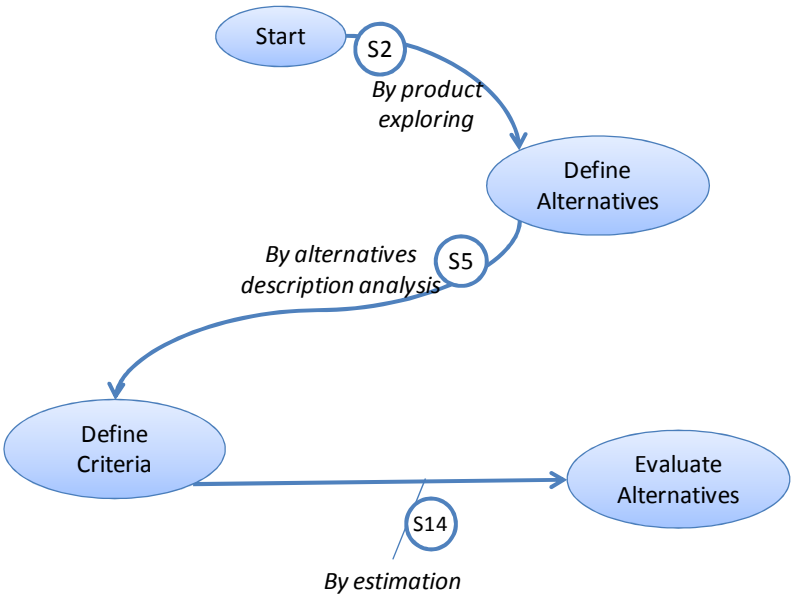


Figure III.2.24. DM Method Line of the Tool Selection in RUP.

The tool selection trajectory includes the following steps. A tool to select is considered as a product (The *By product exploring* strategy is selected). One of the steps in this task is to collect information about tools in order to decide which tool is suitable for the project at hand. The suggested criteria are: (i) tool criteria (features and functions, integration, applicability, extendibility, team support, usability, quality, performance, maturity); (ii) vendor criteria (stability, support availability, training, availability, growth direction); (iii) cost (acquisition cost, implementation cost, maintenance cost). These criteria are based on the tools description (The *By alternatives description analysis* strategy is selected). The RUP proposes to grade each criterion for evaluating candidate tools. The engineer estimates tools according to different scales. Therefore, the evaluation is subjective (The *By estimation* strategy is selected). The recommendations of the RUP methodology stop at this stage. RUP does not contain any method for aggregating evaluations. Table III.2.10. shows the set of the DM components retrieved in the RUP tool selection task.

Table III.2.10. DM Method Line of the RUP Tool Selection: DM Components List.

Section	***Name***
S2	Define alternatives list by product exploring
S5	Define criteria by alternatives description analysis
S14	Evaluate alternatives by estimation

2.5.3. Some Conclusions

As we can see, the cost-value approach provides a more detailed guideline for DM. It allows a complete DM process from the alternatives definition to the decision validation. It contains a

possibility to dynamically adjust an alternatives set. Two examples have different strategies for evaluating alternatives. The tool selection approach is simpler to carry out but it does not contain any decision-making and validation steps (We can see that there is none corresponding DM component in the DM method line).

Both examples are expressed as DM method lines as we have identified a suitable trajectory in the DM method family for each of them. The two DM processes are completely covered by the MADISE approach. Therefore, these examples allow validating the MADISE DM method family.

Moreover, another aspect is highlighted within these examples as the DM method family can contribute to improving existing DM models. For instance, the tool selection approach does not provide any advice for aggregating values, as mentioned above. Therefore, it can be completed by the DM components *Make decision by unique criterion of synthesis* or *Make decision by outranking relation of synthesis* in order to include an aggregation method. In the same way, the cost-value approach may be enhanced by the DM component *Refine alternative list by effective alternatives identification* in order to eliminate dominated alternatives and, in this way, to simplify the AHP method application.

2.6. Conclusion

Our goals in this Chapter were to present the method component meta-model, to provide a means for organizing method component within a family and to show their representation. We based our research on the COMET meta-model which allows modular modeling of method components and families, and on the Map formalism, providing a way for organizing method components within a family and its graphical representation.

The presented structure of method components allows us to identify them, to store them in a methodological repository and to reuse them in the construction of other methods. Thus, they become modules for the construction of new methods. Moreover, they can also be used to enrich existing methods.

Chapter 3: DM Method Family Contextualization

3.1. Introduction

Once a method family is defined, it is necessary to provide guidelines for using the given family in a specific situation. That implies that the context must be defined for the method family and its method components. At the same time, the context is used for specifying the situation of a concrete usage of the method family in order to identify an application method adapted to this situation.

Existing SME approaches consider the notion of context in order to guide the selection of a method component from a repository according to a given situation. They also deal with different kinds of context factors characterizing situations of IS development projects and offer various methodologies for using context.

However, the comparison of context approaches highlights that they address several aspects of context and there is no approach that considers all possible characteristics. There is confusion between the context of a project and the context of method components. Moreover, they do not help in the context characteristics specification.

In this Chapter, we give a brief overview of the context definition in general and, particularly, in the SME field. We introduce then the representation of the context within method families and offer an approach for contextualizing methods in the IS domain. We apply the proposed approach in order to define the context of the DM method components and DM method family.

3.2. Context Definition

The definition of context is based on the review of the context notion used in different fields and on the specific interpretation of context in the SME approaches.

3.2.1. Cross-Domains Notion of Context

[Bouquet *et al.*, 2003] states that the study of context was started in the 70's. Since then, many different domains in relation with information systems use the notion of context and give various interpretations of it. Context models are multidisciplinary and have been proposed in several areas [Bradley *et al.*, 2005]. For instance, [Dey *et al.*, 2001] defines the notion of context by the information that could be used for characterizing the situation of an entity (person, object or computer), and, more generally, by any element that can influence the IS behavior.

[Rey *et al.*, 2002] foresees the context from four points of view:

- The context must be defined in terms of an object. It means, "There is no context without context".
- The capture of context is not the goal in itself but the captured data must serve a purpose.
- The context is an information space shared by multiple actors (users and systems).
- The context is infinite and varies with the passing of time.

Related to the Information technologies field, the context is represented as a model or ontology. For instance, [Gu *et al.*, 2004] suggests a more detailed vision of context. It describes a formal context model based on ontology for intelligent environments. This context ontology defines a vocabulary for representing knowledge about context in this field. It includes two levels: upper ontology (capturing general context knowledge) and domain-specific ontologies (detailing basic concepts in application to a given domain). [Gu *et al.*, 2004] also specifies a way for modeling context classification, dependency between context elements, and quality of context.

In the field of Knowledge Representation and Reasoning (KRR), which is an area of Artificial Intelligence, two types of the context theory have been proposed: (i) *divide-and-conquer*, which sees context as a way of partitioning a global model of the world into smaller and simpler pieces and (ii) *compose-and-conquer*, which sees context as a local theory of the world in a network of relations with other local theories [Bouquet *et al.*, 2003].

Another term, closely related to the context one, is *context-awareness*. Context awareness is a term originating from pervasive computing, or ubiquitous computing [Schilit *et al.*, 1994]. These systems deal with linking changes in the environment with computer systems, which are otherwise static. Although it is a computer science term, it has also been applied to business theory in relation to business process management issues [Rosemann *et al.*, 2006] [Bessai *et al.*, 2008], to computer science [Bradley *et al.*, 2005], and to service selection [Kirsch Pinheiro *et al.*, 2008]. In these works,

the context is mainly used to solve the problem of lacking flexibility and adaptability within processes.

3.2.2. Context in Situational Method Engineering

In the field of SME, the method component context is studied in different approaches and is represented as: reuse frame [Mirbel, 2008]; interface [Ralyté *et al.*, 2001]; method service context [Guzélian *et al.*, 2007]; contingency factors [Van Slooten *et al.*, 1996] [Harmsen, 1997]; development situation [Karlsson *et al.*, 2004] (detailed in Part I).

- **Reuse frame.** The reuse frame [Mirbel, 2008] includes a reuse situation (which is a set of criteria classified into three dimensions: organizational, technique and human) and reuse intention.
- **Interface.** In [Ralyté *et al.*, 2001], the method chunk context includes domain characteristics (project nature, project domain) and human (actor), process and product ontologies.
- **Method service context.** The method service context [Guzélian *et al.*, 2007] aims at describing the situation in project development for which the method service is suitable and defining the purpose of the service. Its model includes *domain* characteristics (project nature, project domain) and *human* (actor), *process* and *product* ontologies.
- **Contingency factors.** In [Van Slooten *et al.*, 1996], four categories are given: domain characteristics (describing the content of the system), external factors (laws and norms), technical factors (related to the development platform) and human factors (representing the development expertise of people).
- **Development situation.** The development situation model includes a characteristics set [Karlsson *et al.*, 2004].

These approaches foresee different context elements that are the characteristics of method components. The comparison of these five approaches enables to identify height characteristics groups (context elements) which allow us to compare existing context approaches (See Table III.3.1.).

Table III.3.1. Comparison of Approaches dealing with Context in ME Field.

Approach	***Characteristics***							
	Goal/ Intention	**Organiza-tional**	**Technical**	**Human**	**Domain**	**External**	**Process**	**Product**
Reuse Frame	X	X	X	X				
Interface	X							X
Method service context				X	X		X	X
Contingency factors			X	X	X	X		
Development situation	Not specified							

The reviewed SME literature shows several drawbacks in reference to the context definition:

- The context notion is not clarified concerning its relationships to methods and method components.
- There is confusion between two aspects of the method context definition: the definition of the context in which a method can be applied and the definition of the context related to a given project.
- Typologies of possible context characteristics are not homogeneous.
- The existing SME literature does not provide means for specifying context characteristics for a given method.

Our approach aims at improving the context definition within the SME field by dealing with these drawbacks.

3.3. Context Representation in the Method Family Model

In this section, we detail the proposed context meta-model, the possible levels of granularity of context, the correspondence between the method context and the project one, and, finally, representation of context within the map model.

3.3.1. Context Meta-Model

In our proposal, we describe the context as a set of characteristics. These characteristics describe situations of a method application. The detailed context model is presented at Figure III.3.2.

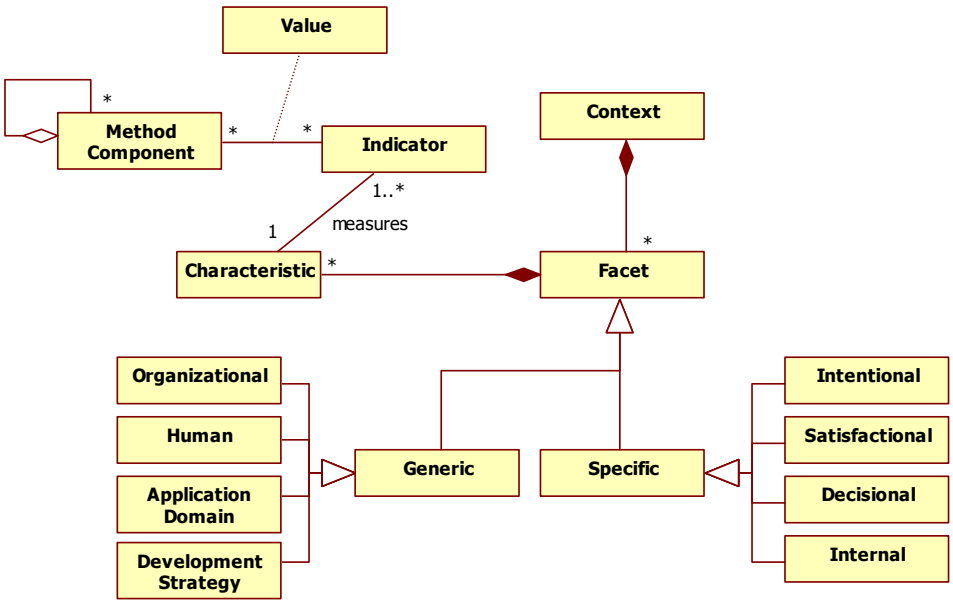


Figure III.3.2. Context Meta-Model.

The central element of this model is the characteristic concept. A set of characteristics constitute the context. *Context characteristics* indicate specific conditions to use the component. Characteristics are organized into *facets* for better representation and comprehension. We distinguish two types of characteristics (and consequently two types of facets): *generic* and *specific*. The first ones are common for most IS engineering projects; the latter ones vary from one project to another. To distinguish between them is important because of their different identification approaches. The context characteristics set is defined for a method component. Therefore, each method component is described by the valuations of these characteristics (*value*). Different context characteristics and the associated indicators are described in Section 3.4.

3.3.2. Context Granularity

We propose to consider the context granularity at several levels: the method family and the method component ones (See Figure III.3.1.). Each method family is available in a given context. As a method is composed of some components, each of them can be also described by specifying its context. Therefore, the method context is an aggregation of contexts associated to its components.

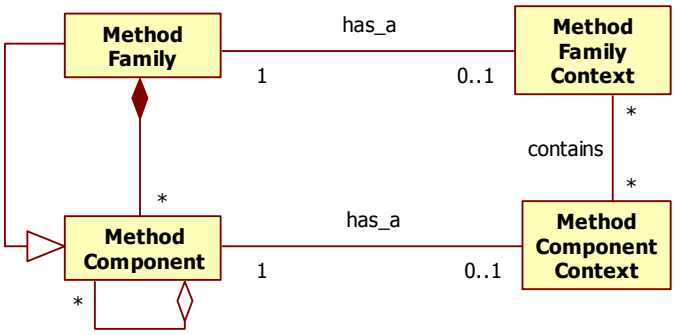


Figure III.3.1. Context Granularity.

3.3.3. Method Components' Context and Project Context

The part of the method family meta-model corresponding to the context is shown at Figure III.3.4.

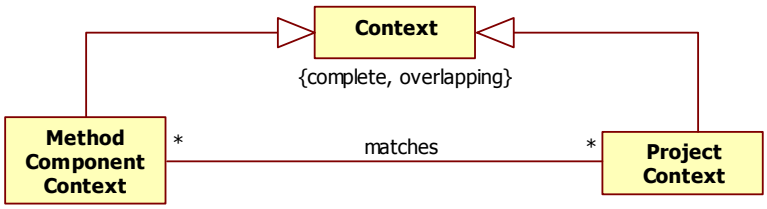


Figure III.3.4. Method Component and Project Context.

The context describes conditions in which method components are applied. The context is represented using taxonomy of characteristics, such as cost, time, etc. For both method component and project situation, the context must be specified. The two contexts are specified on the same basis as they inherit from the ***context*** (the {complete, overlapping} inheritance relationship). This allows a better matching between the situation and components for the configuration (the *matches* relationship).

The ***method component context*** characterizes all possible situations in which this component may be applied (by the *is_characterized_by* association). We associate the description of the context to each method component. The information about the component context is further used for configuring application methods (method lines).

The ***project context*** includes all characteristics of the situation at hand, or in other words of the variation point. The characterization of the project context allows configuring a method line from the existing method family. Thereby, the obtained method line matches the situation at hand.

3.3.4. Context Representation within Maps

The usual navigation on the map is based on a set of arguments which are not always sufficiently detailed and which do not allow to select the right components in an obvious manner. We suggest to describe method components and, therefore, different elements of map with the use of various quantitative and qualitative characteristics. These criteria become a prerequisite to have a better guidance.

Thus, we associate a set of characteristics to each level of method component granularity: from the method family to the simple atomic components. As the method family is expressed with Map, we must also define the corresponding characteristics for the DM map in order to provide means for the further DM method family configuration. We have defined the context as a part of method components in Chapter 2 of this Part. The context is composed of a set of characteristics measured with different indicators. Method components are evaluated according to these characteristics in order to define the situations in which their application is suitable.

In the same manner, we attribute characteristics to the map sections instead of the different guidelines proposed by the Map model (See Figure III.3.5.). Context characteristics enrich the existing guidelines and make them more formalized in order to provide simpler configuration of the method family.

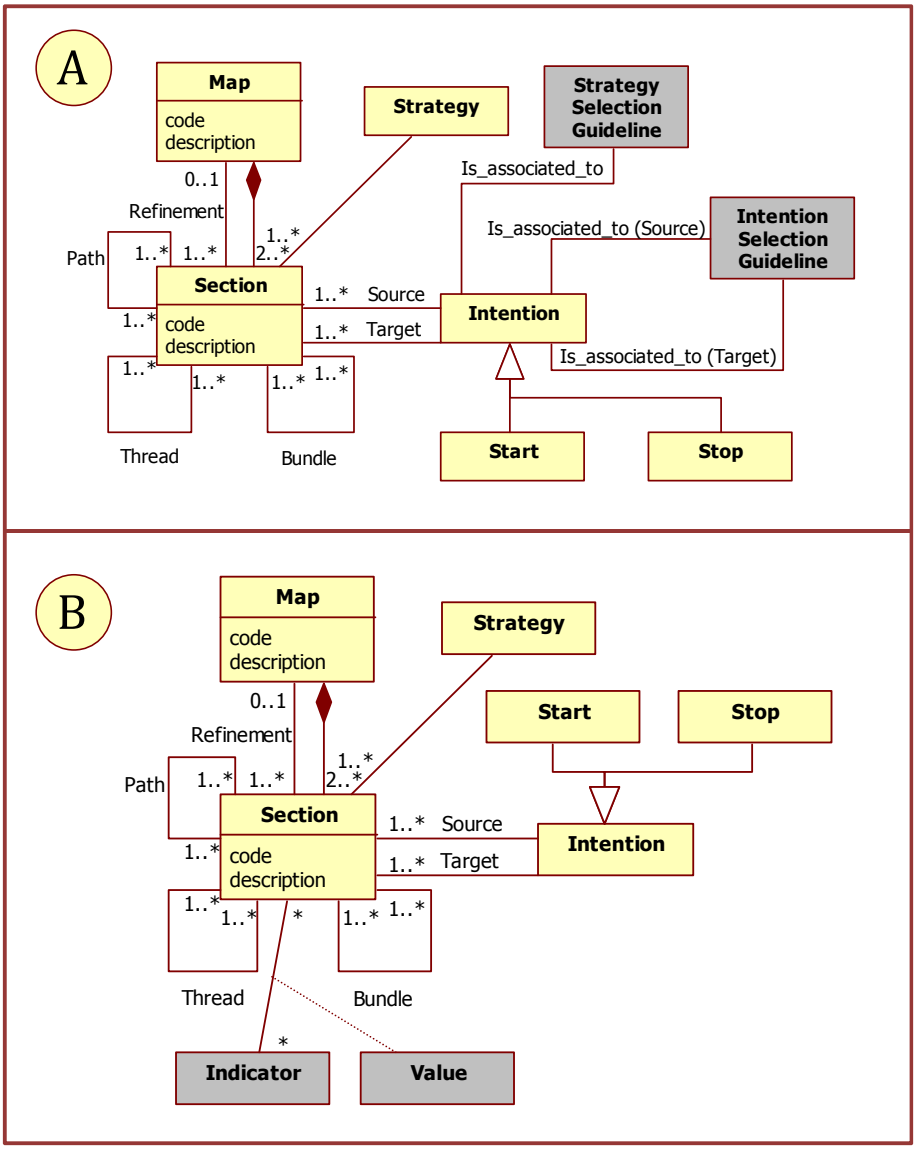


Figure III.3.5. MAP Basic Model (A) and MAP Enhanced Model (B).

3.4. Typology of Context Characteristics

In the following, we describe different context characteristics by facets.

3.4.1. Generic Characteristics

In order to establish the typology of generic characteristics we have used IS development project characteristics [Kornyshova *et al.*, 2007c]. In this work, a project characteristics typology is proposed in order to guide method components retrieval and to prioritize the selected components.

The suggested typology of context characteristics covers essential aspects of IS engineering projects. Based on [Mirbel *et al.,* 2006], [Van Slooten *et al.*, 1996] and [Kornyshova *et al.,* 2007c], it includes four facets: organizational, human, application domain, and development strategy.

The *organizational* facet (Table III.3.2.) highlights organizational aspects of IS project development. For instance, the *Management Commitment* characteristic represents the management team involvement in the project. Possible values for this characteristic are Low, Normal and High (i.e. a High value means a high involvement and so on).

Table III.3.2. Organizational Facet Characteristics.

Characteristic	Indicator	Type	Value domain
Management commitment	Management commitment degree	Quantitative	3-grade scale
		Qualitative	ENUM:{low, normal, high}
Importance	Importance degree	Quantitative	3-grade scale
		Qualitative	ENUM:{low, normal, high}
Impact	Impact degree	Quantitative	3-grade scale
		Qualitative	ENUM:{low, normal, high}
Time pressure	Time pressure degree	Quantitative	3-grade scale
		Qualitative	ENUM:{low, normal, high}
Shortage of resources	Shortage of resources degree	Quantitative	3-grade scale
		Qualitative	ENUM:{low, normal, high}
Level of innovation	Level of innovation degree	Quantitative	3-grade scale
		Qualitative	ENUM:{low, normal, high}
Size	Size	Quantitative	3-grade scale
		Qualitative	ENUM:{low, normal, high}
Cost	Cost	Quantitative	REAL
		Qualitative	ENUM:{low, normal, high}
Nature of limited resources	Nature of limited resources	Qualitative	ENUM:{financial, human, temporal, informational }
Innovation nature	Innovation nature	Qualitative	ENUM:{business innovation, technology innovation}
Duration	Duration	Quantitative	REAL

The *human* facet (Table III.3.3.) describes the qualities of persons involved in IS project development. For example, the *User involvement* characteristic represents the kind of participation of the users in the project. Its values may be real or virtual.

Table III.3.3. Human Facet Characteristics.

Characteristic	Indicator	Type	Value domain
Resistance	Resistance degree	Quantitative	3-grade scale
		Qualitative	ENUM:{low, normal, high}
Conflict	Conflict degree	Quantitative	3-grade scale
		Qualitative	ENUM:{low, normal, high}
Expertise	Expertise degree	Quantitative	3-grade scale
		Qualitative	ENUM:{low, normal, high}
Clarity	Clarity degree	Quantitative	3-grade scale
		Qualitative	ENUM:{low, normal, high}
Stability	Stability degree	Quantitative	3-grade scale
		Qualitative	ENUM:{low, normal, high}
Expert role	Expert role	Qualitative	ENUM:{tester, developer, designer, analyst}
User involvement	User involvement	Qualitative	ENUM:{real, virtual}
Stakeholder number	Stakeholder number	Quantitative	NUMBER

The *application domain* facet (Table III.3.4.) includes indicators characterizing the domain of IS project. For instance, the *Application type* characteristic deals with the different kinds of projects according to the organization structure and can have the following values: intra-organization application, inter-organization application, organization-customer application.

Table III.3.4. Application Domain Facet Characteristics.

Characteristic	Indicator	Type	Value domain
Formality	Formality degree	Quantitative	3-grade scale
		Qualitative	ENUM:{low, normal, high}
Relationships	Relationships degree	Quantitative	3-grade scale
		Qualitative	ENUM:{low, normal, high}
Dependency	Dependency degree	Quantitative	3-grade scale
		Qualitative	ENUM:{low, normal, high}
Complexity	Complexity degree	Quantitative	3-grade scale
		Qualitative	ENUM:{low, normal, high}
Repetitiveness	Repetitiveness degree	Quantitative	3-grade scale
		Qualitative	ENUM:{low, normal, high}
Variability	Variability degree	Quantitative	3-grade scale
		Qualitative	ENUM:{low, normal, high}
Application type	Application type	Qualitative	ENUM:{intra-organization, inter-organization, organization-customer }
Application technology	Application technology	Qualitative	ENUM:{application to develop includes a database, application to develop is distributed, application to develop includes a GUI}
Dividing project	Dividing project	Qualitative	ENUM:{one single system, establishing

			system-oriented subprojects, establishing process-oriented subprojects, establishing hybrid subprojects}
Variable artefacts	Variable artefacts	Qualitative	ENUM:{organisational, human, application domain, and development strategy}

The *development strategy* facet (Table III.3.5.) gathers indicators about different characteristics of development strategy. For instance, the *Source system* characteristic represents the origin of the reused elements that may be code, functional domain or interface.

Table III.3.5. Development Strategy Facet Characteristics.

Characteristic	Indicator	Type	Value domain
Source system	Source system	Qualitative	ENUM:{code reuse, functional domain reuse, interface reuse}
Project organization	Project organization	Qualitative	ENUM:{standard, adapted}
Development strategy	Development strategy	Qualitative	ENUM:{outsourcing, iterative, prototyping, phase-wise, tile-wise}
Realization strategy	Realization strategy	Qualitative	ENUM:{at once, incremental, concurrent, overlapping}
Delivery strategy	Delivery strategy	Qualitative	ENUM:{at once, incremental, evolutionary}
Tracing project	Tracing project	Qualitative	ENUM:{weak, strong}
Goal number	Goal number	Quantitative	NUMBER
		Qualitative	ENUM:{one goal, multi-goals}

3.4.2. Specific characteristics

The identification of specific characteristics is based on the particular method description or representation. The method engineer defines them by analyzing different aspects. Different elements of methods are analyzed in order to elicit a set of characteristics allowing distinguishing the specific context of the method application.

For instance, for methods expressed with maps, specific characteristics could be organized into four facets: intentional, satisfaction, decisional and internal, like in [Harmsen, 1997]. The *intentional* facet concerns the method intentions. The *satisfaction* facet indicates the satisfaction degree that the engineer has about the method application results. The *decisional* facet arises from a decision-making process in the method. The *internal* facet concerns the known criteria associated with the specific project management. For the specific map characteristics see Table III.3.6.

Table III.3.6. Specific Map Characteristics.

Characteristic	Indicator	Type	Value domain
Goal satisfaction degree	Goal satisfaction degree	Quantative	3-grade scale.
		Qualitative	ENUM:{low, normal, high}
Goal achievement degree	Goal achievement degree	Quantative	3-grade scale
		Qualitative	ENUM:{low, normal, high}
Section satisfaction degree	Section satisfaction degree	Quantative	3-grade scale
		Qualitative	ENUM:{low, normal, high}
Section completeness degree	Section completeness degree	Quantative	3-grade scale
		Qualitative	ENUM:{low, normal, high}

Table III.3.7 shows the correspondence between the proposed typology and the existing context elements (analyzed in the previous section). We can make some remarks to compare them:

- Our typology covers all existing elements.
- We propose to identify more precisely process and product characteristics using our approach instead of using product and process as context characteristics directly.
- We add decisional characteristics that are not presented in the existing typologies.

Table III.3.7. Correspondence between the Proposed Typology and Existing Context Elements.

Proposed Typology	Context Elements (cf. Table III.3.1.)
Organizational	Organizational
Human	Human
Application domain	Domain
Development strategy	Technical
Intentional	Goal/ Intention
Satisfaction	External, Process, Product
Decisional	Process, Product
Internal	Technical, Process, Product

3.4.3. General View of the Context Typology

Our typology indicates the main characteristics that can be defined in function of a given situation. It can be completed if new characteristics arise. Figure II.3.3 illustrates the obtained characteristics typology as an ontology, like in [Gu *et al*., 2004].

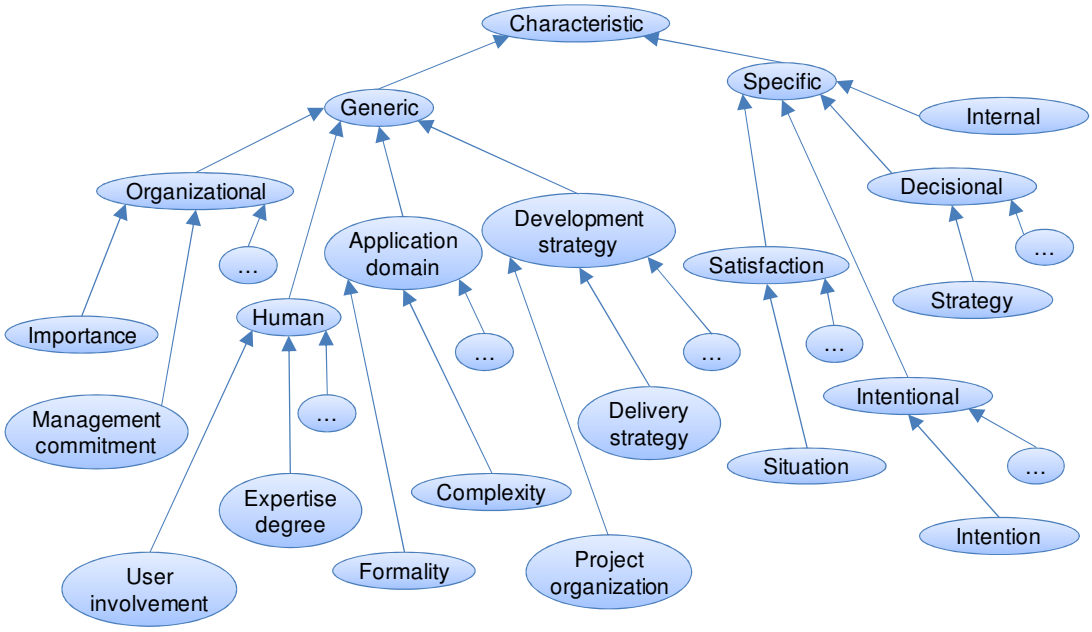


Figure III.3.3. Characteristics Ontology.

This ontology allows representing the characteristic typology as a hierarchy, which contains the specialization links, and is more easily understandable.

3.5. Method Contextualization Approach

If our first goal was to enhance the definition of the context of IS development method for the further configuration of application methods from a method family according to a given situation; our second goal is to provide an approach to define the context for a given method family and its components (the contextualization approach).

We use the Map model for modeling the contextualization approach (See Figure III.3.6.). This approach includes two possible ways to define the context: top-down or bottom-up. By the top-down approach, the engineer defines the method family context and then instantiate it for each method component. By the bottom-up approach, the engineer specifies the contexts of all method components and assemblies them into the method family context. Both method family and method component contexts can be defined following two strategies: *By deduction* and *By generation*. It depends on the characteristic type. The generic characteristics are *deduced* from the generic context typology and the specific ones are *generated* from method description. These strategies could be applied as many times as possible characteristics exist.

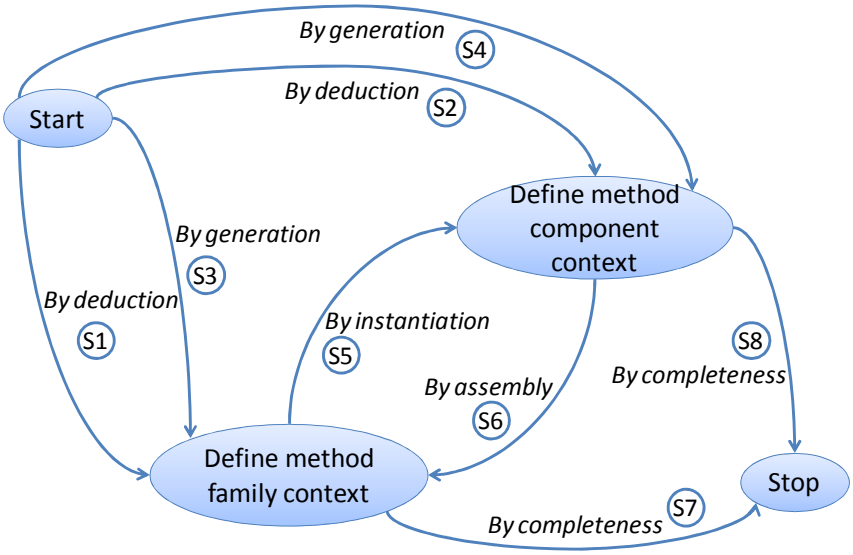


Figure III.2.6. Contextualization Map.

This MAP has two main intentions: *Define method family context* and *Define method component context*. The achievement of these intentions implies the definition of the context characteristics set for method family or for method components respectively. The definition of method components contexts includes also the attribution of values to the defined characteristics.

The contextualization Map includes eight sections, as shown in Table III.3.8.

Table III.3.8. Contextualization Map Sections.

Section	***Signature***
S_1	<(Method Family = Defined), Define method context by deduction>
S_2	<(Method Components = Defined), Define method component context by deduction>
S_3	<(Method Family = Defined), Define method context by generation>
S_4	<(Method Components = Defined), Define method component context by generation>
S_5	<(Method Family Context = Defined), Define method component context by instantiation>
S_6	<(Method Components Context = Defined), Define method context by assembly>
S_7	<(Method Family Context = Defined), Stop by completeness>
S_8	<(Method Components Context = Defined), Stop by completeness>

All these sections are explained below. Operators are defined for each section in order to indicate how to proceed for carrying out its execution.

<(Method Family = Defined), Define method context by deduction>. The generic characteristics deduction is based on the context typology. This section gives a selection of characteristics carried out by the IS method engineer. The result of this strategy is a sub-set of generic characteristics available for a given project. The corresponding operator is:

Select Context Characteristic ()

<(Method Components = Defined), Define method component context by deduction>. This section includes the selection of characteristics form generic typology like the previous one and includes furthermore the attribution of values to these characteristics. The result of this strategy is a sub-set of generic characteristics available for a given project with corresponding values. Two following operators are applied consecutively:

Select Context Characteristic ()
Attribute a Value to Context Characteristic ()

<(Method Family = Defined), Define method context by generation>. The specific characteristics generation is based on the method description. The method engineer defines them by analyzing different aspects which are organized into four facets: intentional, satisfaction, decisional and internal. This section includes four operators. Each of the following operators is applied depending on the corresponding characteristic's facet:

Analyze Goal ()
Measure Satisfaction ()
Analyze Argumentation ()
Measure Characteristics ()

<(Method Components = Defined), Define method component context by generation>. The definition of specific characteristics for method components context is the same as for method context (the previous section) but also requires the attribution of characteristics values. This section uses the same four operators and adds another one that deals with the attribution of values to the characteristics. This last one is applied after each of the first four operators for defining concrete values of the identified specific characteristics.

Analyze Goal () [for intentional facet]
Measure Satisfaction () [for satisfaction facet]
Analyze Argumentation () [for decisional facet]
Measure Characteristics () [for internal facet]
Attribute a Value to Context Characteristic () [for all facets]

<(Method Family Context = Defined), Define method component context by instantiation>. The context characteristics instantiation is common for both characteristics types and is applied in the top-down approach. This section allows defining a sub-set of generic and specific method characteristics with an associated value for each method component separately. Several functions may be applied (sum, maximum, minimum, average, weighted sum, and so on) for attributing values to the method family context. This section contains two operators applied consecutively:

Retain Context Characteristic ()
Attribute a Value to Context Characteristic ()

<(Method Components Context = Defined), Define method context by assembly>. In the case of the bottom-up approach, the strategy *By assembly* follows the definition of the method component context *By deduction* or *By generation*. The method engineer groups method components characteristics together. As a result, the method family context includes all characteristics of its components contexts. The application of this strategy allows also defining characteristics' values for the method family context. This section is carried out by the following operator:

Group Characteristics ()
Attribute a Value to Context Characteristic ()

<(Method Family Context = Defined), Stop by completeness> and **<(Method Components Context = Defined), Stop by completeness>.** These sections are the same in both top-down and bottom-up approaches and include verification of completeness and coherence of the described context. The associated operator is:

Verify Context Completeness ()

All these operators are resumed in the Table III.3.9.

Table III.3.9. Operators' Description.

Operators	***Description***
Select Context Characteristic ()	Helps to select context characteristics which are pertinent for a given method family of method component.
Attribute a Value to Context Characteristic ()	For each characteristic selected, a value corresponding to the method family or to the method component context has to be defined.
Analyze Goal ()	Helps to define the characteristics of the intentional facet which concerns the intentions (goals) of a method family of a method component.
Measure Satisfaction ()	Helps to measure the satisfaction degree on the results obtained by the engineer for a method family of a method component
Analyze Argumentation ()	The decisional facet needs this operator in order to describe a decision-making situation with the definition of the arguments to take into account in the DM process for a method family of a method component
Measure Characteristics ()	This operator is used to give values to the characteristics associated with the a method family of a method component.
Retain Context Characteristic ()	Helps to define a subset of characteristics (generic or specific characteristics) and to give them values adapted to a method component.
Group Characteristics ()	This operator allows to group all the characteristics together in the same set which corresponds to the context of a method family.
Verify Context Completeness ()	Helps to study the completeness and the coherency of the context set of characteristics.

3.6. Defining Context for the MADISE Method Family

In this section, we apply the contextualization approach in order to define the context of DM method components and DM method family. This represents an example of the contextualization approach application carried out for the purpose of our research. However, it can be used in other ISE context by selecting the suitable contextualization operators.

3.6.1. Contextualization Process

We have selected the top-down approach of the contextualization approach. It guides through the definition of the method family characteristics before the definition of method component characteristics. Figure III.2.7. shows the path used in the navigation through the contextualization map.

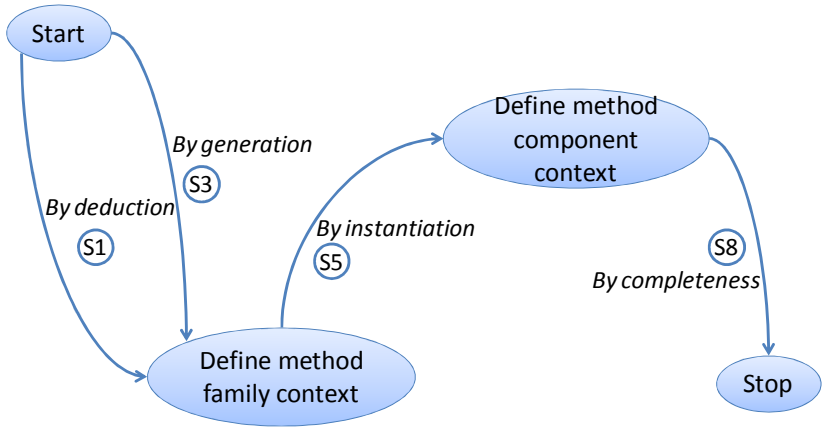


Figure III.2.7. Contextualization Map for the DM Method Family.

This path contains four sections of the Contextualization Map that we represent within the three following steps:

1. Definition of the method family context (S_1 and S_3),
2. Definition of the method components contexts (S_5), and
3. Verification of the process completeness (S_8).

3.6.1.1. Definition of the Method Family Context

The first step (*S_1, S_3: Definition of Method Family Context*) contains the execution of two sections of the Contextualization Map: *S_1* and *S_3*.

Definition of the generic characteristics *(S_1).* We use the characteristics presented above as generic characteristics to specify the context of the DM method family. We have applied the operator *Select Context Characteristic ()* in order to define a sub-set of generic context characteristics. The selected characteristics are described in the following.

- ***Time pressure.*** Time pressure indicates the urgency of decision-making. If a decision must be made as soon as possible, then, generally, there is no time to carry out a detailed analysis of the situation and to apply a complicate DM method. For this reason, we characterize each component according to its suitability to make decisions in the condition of time pressure. The related indicator is the time pressure degree. It could have three values: {low, normal, high}. The low value of a component shows that this component requires a long time for its realization.
- ***Shortage of resources.*** The shortage of resource is related to the availability of resources required to perform an action. In the case of the DM components, the matter is about financial and informational resources. The financial resources relate to the possibility to finance the detailed DM; the informational ones show the data available for decision-making. Both related indicators (Shortage of financial resources degree and Shortage of informational resources degree) can be {low, normal, high}. The high value means that the given method component is adapted for shorted resources.
- ***Duration.*** The duration indicates the time needed for executing a component. We have also chosen to measure this characteristic with an indicator having quantitative values according to five levels scale, as the real duration of DM activities is variable. This characteristic is related to the cost one. Obviously, activities that require more time to be executed are more expensive. For this reason, we do not consider the cost indicator in this thesis. However, it could be taken into account when DM activities are outsourced and, consequently, are bought by the company.
- ***Expertise.*** The expertise characteristic relates to the level of required knowledge in the DM field in order to perform the given activity. We suggest measuring this characteristic with the qualitative three levels scale {low, normal, high}. The low expertise degree shows that the given component may be applied by an engineer having a low level of DM knowledge.
- ***Expert role.*** As detailed in Chapter 2 of Part I, the possible roles in DM activities are stakeholder, IS engineer, and DM staff. Thus, we adapt the expert role characteristic to our case, and the corresponding qualitative indicator has the following values {stakeholder, IS engineer, DM staff}
- ***User involvement.*** Several DM activities, as, for instance, the preference elicitation, require the user's involvement. Other ones could be executed automatically by a machine (like calculation of the aggregated values) without user's participation. We introduce the user involvement characteristic in order to differentiate DM method components according to the type of user's participation with two possible values {required; not required}.
- ***Complexity.*** The complexity characteristic relates to the level of difficulty of the method components application. The complexity of the DM method components varies from very simple to very complex ones depending on the associated type of guideline. A complex component may require more time to apply it. We suggest measuring this characteristic with the complexity degree indicator described by a quantitative value depending on the nature and the number of the included guidelines.

These characteristics are summed up in Table III.3.10.

Table III.3.10. Generic Characteristics Selected for DM Method Family.

Characteristics	Indicator	Type	Value domain
Time pressure	Time pressure degree	Qualitative	ENUM:{low, normal, high}
Shortage of resources	Shortage of resources degree	Qualitative	ENUM:{low, normal, high}
Duration	Duration	Quantitative	5-levels scale
Expertise	Expertise degree	Qualitative	ENUM:{low, normal, high}
Expert role	Expert role	Qualitative	ENUM:{stakeholder, IS engineer, DM staff}
User involvement	User involvement	Qualitative	ENUM:{required; not required}
Complexity	Complexity degree	Quantitative	Real

Definition of the specific characteristics *(S_3).* We use also *specific* context characteristics appropriated to the DM field. This specific context is depicted by analyzing characteristics of DM methods. Thus, we have defined a set of characteristics related to the main DM concepts (problem, alternatives, and criteria), to the usage of DM methods in practices, and also the situations, intentions, and strategies embedded to each component.

- ***Problem type.*** This characteristic shows whether a component could deal with different kinds of the decision problem {choice, ranking, sorting, description}.
- ***Alternative number.*** It describes the number of alternatives {great, medium, small} that can be considered using the method.
- ***Alternative compatibility.*** This characteristic is introduced for considering components that have the ability to take into account incompatibilities and conflicts between alternatives. This can be {yes, no}.
- ***Criteria data type.*** Different DM methods are applicable with predefined data type that can be {quantitative, qualitative, mixed, fuzzy}.
- ***Criteria measure.*** The criteria measure scale shows the ability of methods to deal with different scales which are {nominal, ordinal, interval, ratio, absolute, mixed}.
- ***Criteria weighting.*** This characteristic shows the manner of the decision criteria definition (either without weighting, with weighting and interdependencies, or simple weighting) which give the following values {no, simple, interdependence}.
- ***Criteria compensation.*** Different DM components can consider the criteria compensation {yes, no}.
- ***Formalism.*** The formalism of a DM method component characterizes the nature of the associated guideline. It can be {formal, semi-formal, informal}.
- ***Easiness.*** The easiness indicates the facility of the component usage. This characteristic is measured with the easiness degree indicator described by a qualitative three levels scale {easy, medium, difficult}.
- ***Skills.*** This characteristic is related to the decision maker's skill {week, medium, strong} required for the DM method component execution.
- ***Problem scale.*** The problem scale relates to different characteristics presented in Part I –

Chapter 2 when describing decision types in ISE (See Figure I.2.7.). The decisions positioned at the right side of the figure correspond to the high problem scale degree, and the decisions at the left side to the low degree {low, medium, high}.

- ***Tool.*** The tool characteristic indicates if a tool is available to support a method. The possible values are {yes, no}.
- ***Intention.*** The intention characteristic describes the main goal of the DM component. It is represented by a text.
- ***Situation.*** The situation characterizes the DM elements required for the application of a DM method component. It contains different DM objects and formalized as a text.
- ***Strategy.*** The strategy characteristic describes the strategy used for executing the DM component. It is represented by a text.

These characteristics are shown in Table III.3.11.

Table III.3.11. Specific Characteristics Defined for the DM Method Family.

Characteristics	Indicator	Type	Value domain
Problem type	Problem type	Qualitative	ENUM:{choice, ranking, sorting, description}
Alternatives number	Alternatives number	Qualitative	ENUM:{great, medium, small}
Alternative compatibility	Alternative compatibility degree	Qualitative	ENUM:{yes, no}
Criteria data type	Criteria data type	Qualitative	ENUM:{quantitative, qualitative, mixed, fuzzy}
Criteria measure scale	Criteria measure scale	Qualitative	ENUM:{nominal, ordinal, interval, ratio, absolute, mixed}
Criteria weighting	Criteria weighting	Qualitative	ENUM:{no, simple, interdependence}
Criteria compensation	Criteria compensation	Qualitative	ENUM:{yes, no}
Formalism	Formalism degree	Qualitative	ENUM:{informal, semi-formal, formal}
Easiness	Easiness degree	Qualitative	ENUM:{easy, medium, difficult}
Skills	Skills degree	Qualitative	ENUM:{week, medium, strong}
Problem scale	Problem scale	Qualitative	ENUM:{low, medium, high}
Tool	Tool	Qualitative	EMUN:{yes, no}
Intention	Intention	Qualitative	String
Situation	Situation	Qualitative	String
Strategy	Problem type	Qualitative	String

3.6.1.2. Definition of Method Components Context

On the second step (S_5: *Definition of Method Components Context*), the method family context defined at the previous step is instantiated for each component. A value is affected to each characteristic in order to help the case process execution guidance. The following operators are applied to each method characteristic: *Retain Context Characteristic ()* and *Attribute a Value to Context Characteristic ().* Table III.3.12. illustrates the obtained context of the DM method

components of the highest level of the DM method family according to three following indicators: complexity degree, expertise degree, and intention.

Table III.3.12. Example of DM Method Components Evaluations

Section	*DM Method Component*	*Complexity degree*	*Expertise degree*	*Intention*
S1	<(DM Object defined), Define alternatives by process exploring>	2	2	Define alternatives
S2	<(DM Object defined), Define alternatives by product exploring>	2	2	Define alternatives
S3	<(Alternatives defined), Define alternatives by elimination>	1	1	Define alternatives
S4	<(Alternatives defined), Define alternatives by addition>	1	1	Define alternatives
S5	<(Alternatives defined), Define criteria by alternatives description analysis>	2	1	Define criteria
S6	<(Alternatives defined), Define criteria by consequences analysis>	2	1	Define criteria
S7	<(Alternatives defined), Define criteria by goal analysis>	1	1	Define criteria
S8	<(Alternatives defined), Define criteria by predefined list exploring>	1	1	Define criteria
S9	<(Criteria defined), Define criteria by elimination>	1	2	Define criteria
S10	<(Criteria defined), Define criteria by addition>	1	2	Define criteria
S11	<(Criteria defined), Define criteria by weighting>	2	1	Define criteria
S12	<(Alternatives defined), Evaluate alternatives by preferences analysis>	2	1	Evaluate alternatives
S13	<(Alternatives defined, Criteria defined), Evaluate alternatives by measuring>	2	1	Evaluate alternatives
S14	<(Alternatives defined, Criteria defined), Evaluate alternatives by estimation>	2	1	Evaluate alternatives
S15	<(Alternatives defined, Criteria defined), Evaluate alternatives by preferences analysis>	2	1	Evaluate alternatives
S16	<(Alternatives evaluated, Criteria defined), Evaluate alternatives by preferences analysis>	2	2	Evaluate alternatives
S17	<(Alternatives evaluated), Refine alternative list by acceptable alternatives identification>	2	2	Refine alternative list
S18	<(Alternatives evaluated), Refine alternative list by effective alternatives identification>	2	2	Refine alternative list
S19	<(Alternatives defined), Make decision by using arguments>	1	2	Make decision
S20	<(Alternatives defined), Make decision by “From scratch” strategy>	1	3	Make decision
S21	<(Alternatives evaluated), Make decision by unique criterion of synthesis>	2	2	Make decision
S22	<(Alternatives evaluated), Make decision by outranking relation of synthesis>	3	3	Make decision
S23	<(Alternatives evaluated), Make decision by expertise>	1	3	Make decision
S24	<(Decision made), Prescribe decision by validation>	1	2	Prescribe decision
S25	<(Alternatives defined), Prescribe decision by validation>	1	3	Prescribe decision

For instance, the <(Criteria defined), Define criteria by weighting> DM method component is useful in the situation characterized by the level 2 of complexity (normal), requiring the level 1 of expertise (low), and when the goal is to define the relative importance of criteria.

3.6.1.3. Verification of the context completeness

The engineer has decided that the identified context characteristics are sufficient by applying the operator *Verify Context Completeness ()*.

3.7. Conclusion

This chapter offers a contextualization approach that helps engineers to define the method components context, easily. This process can be applied in different IS engineering situations such as the selection of a component for enhancing an existing IS engineering method (for instance, extension-based approaches) or a selection of several components for constructing a new one (for instance, assembly-based approaches).

Our proposal is (i) to give a rigorous definition of a method component context and (ii) to offer a contextualization process that help engineers to define the method components context with ease.

- **Rigorous definition of a method component context.** We have studied the literature in order to define the criteria that may be used to characterize the situation in which method components may be used. This leads us to define a typology of characteristics that we have structured in different facets (each considering a special view of a project). We then related these characteristics to the method component concepts.
- **Contextualization process.** We have identified two possible ways to use these characteristics for defining context (the top-down and the bottom-up approaches) in order to propose a contextualization process that may be adapted to several situations. We modeled this process with the MAP formalism in order to keep a high level of flexibility in the process utilization.

With relation to the DM method family, the main contributions of this contextualization approach are the following:

- a more simple navigation through the DM method family: characteristic values help the engineer to choose his path through the family and its components;
- a better 'context-aware' selection of DM method components: the context is taken into account in the characteristic values.

The contextualization approach was also applied to two other case studies: scenario conceptualization and project portfolio management [Kornyshova *et al* , 2011a].

Chapter 4: DM Method Family Configuration

4.1. Introduction

Once a method family is constructed and its context defined, we can say that it is ready to be used in different situations. However, in order to proceed to the method execution, an application method must be produced from the method family so that the obtained method is suitable to the given situation. We call this process the method family configuration, and the method obtained from the method family by configuration is called an application method (or a method line).

The configuration process is based on the ability to derive an application method from common characteristics in a repeatable manner following specific criteria. As in product families [Deelstra et al., 2004], each method family member derives its architecture from the method family shared architecture, and select and configure a subset of the method family components.

In this thesis, method families are represented with the intentional model Map. This formalism allows specifying models in a flexible way by focusing on the process intentions, and on the various ways to achieve each of these intentions. A map has a teleological nature as it takes into account the teleological behavior of a process execution. It describes the intentions (goals, objectives) associated to a result that the designer wants to achieve. In that way, the intention-oriented models presuppose decisions which concern different possibilities to carry out the given process. Each decision is a variation point. We suggest to foresee these models as method families as they contain common and variable elements. Each method family can be customized in a given project into an application method. The application method customization is realized by selecting a particular method component in each variation point. Our proposal is to use different DM techniques for guiding this selection.

Therefore, we foresee the method family configuration problem as a decision-making (DM) problem. In our case, alternatives are candidate method components in each variation point, and criteria are indicators characterizing the given project situation.

This Chapter details the method configuration process in general and offers the application of this process to the case of the DM method family.

4.2. Method Family Configuration

In this section, we propose the generic process of the method family configuration. It includes the general view of this process and the detailed description of the three main strategies of configuration using an example from the requirement engineering field dealing with the scenario elicitation activities.

4.2.1. General View of the Method Family Configuration Process

The method family configuration process is an intention-oriented process that we represent with the Map model. The method family configuration map is given at Figure III.4.1.

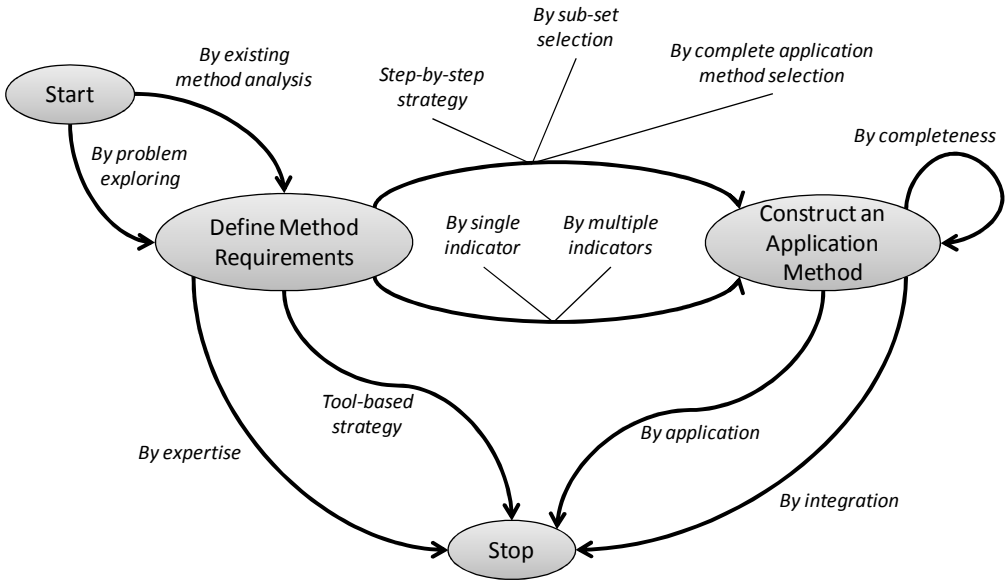


Figure III.4.1. Method Family Configuration Map.

This map shows the main intentions belonging to the process of the method family configuration: *Define Method Requirements* and *Construct an Application Method*.

4.2.1.1. Define Method Requirements

At this stage, a set of requirements for configuring a method family is defined. The requirements to identify must allow selecting the appropriate method components and constructing an application method. Thus, these requirements have to be sufficient to carry out the configuration process realized at the next stage (the *Construct an application method* intention). A method family is formalized in the intention-oriented manner using the map model. This kind of modeling offers the possibility to identify the intentions and situations associated to all components. Based on the intentions, the engineer can define if a method component is suitable for his goals; and the situation is used for identifying the input of each component. In addition, our vision of the map model enhanced with the notion of indicator allows to access to the context associated to each component. In this manner, at this stage the engineer defines a set of elements which are essential for the further configuration of the method family into an application method. The elements to specify at this stage are the intention, situation, and the values of the context indicators.

Two possible strategies to attain this intention are available: *by problem exploring* and *by existing method analysis*.

- ***By problem exploring.*** The engineer begins from zero when he does not have any methodological guideline for the given field.
- ***By existing method analysis.*** In this case, a method for a given field exists but must be improved. This strategy allows analyzing existing methodological elements and to complete them.

Once the method requirements are defined the proper method family configuration could be started.

4.2.1.2. Construct an Application Method

The second step is to use the identified requirements for the guidance. Three kinds of configuring method families are combined with the usage of one or multiple indicators during the configuration.

Three kinds of configuration we suggest are:

- **Step by step** strategy guides the engineer on each of the steps, one by one, to use a method family. In this case, the sub-set of method components (available at the given step) is considered (A'). For instance, the engineer must select between two different strategies allowing to attain the same intention.
- **By sub-set selection** strategy uses indicators to simplify the method family by suppressing some components. This kind of configuration allows to select a sub-set of components (alternatives – A'') from A based of the values of G (criteria set) in a given project. For instance, it can be the selection of method components requiring a low expertise degree.
- **By complete application method selection** strategy helps to select, right from the start of the navigation, between all the possible application methods (or method lines). The first step

is to measure the indicator values for each method line based on values associated to method components which compose this line. Therefore, the alternatives are method lines (A''') and criteria are aggregated values (G'''). For instance, all method components may be measured according to the duration criteria. First, the duration of all possible method lines is calculated (as a sum of its components duration). Then, a method line having the lowest duration is selected.

Moreover, for each of these three kinds of guidance, there are also two separate ways to proceed which are the use of a single indicator or of multiple indicators, as illustrated at Figure III.4.2.

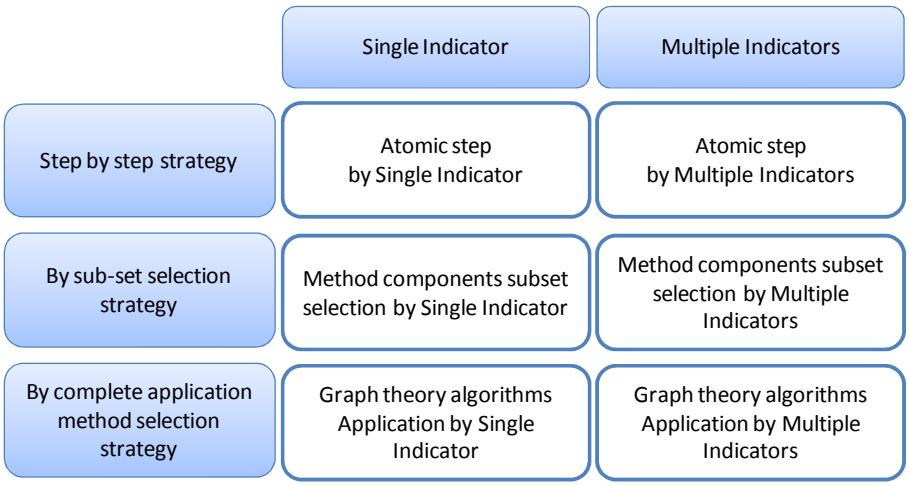


Figure III.4.2. Configuration Types.

The engineer must identify two elements to choose the right guidance. First, he determines his needs: to choose appropriate components, to find an optimal method line, or to carry out a dynamic choice of components. Second, he looks at the number of available indicators and chooses one or several of them. Depending on these choices, he can then select a guidance type adapted to his project case.

By Completeness. Once the first configuration is carried out, the obtained application method must be checked according to its completeness (*By completeness* strategy). The composite component (an application method) issue from the configuration process is compared with the original method family. As this family is represented with a map, the method components, pre-selected during the configuration, are positioned into the map of the method family. For this, the following steps are executed one by one:

- Organize the pre-selected method components within an application method;
- Identify method components which are missing in the obtained application method and, consequently could be added to the application method in order to complete it;
- Identify method components which are concurrent to each other in the obtained application method;

- If necessarily, arbitrate between two or more method components using the context characteristics.

Therefore, the usage of the map model allows assembling the selected components, adding other components if it is necessary for completing the application method, or eliminating excessive ones.

This strategy is not applicable when the configuration is made using the step-by-step strategy as this one allows constructing a complete and coherent application method.

4.2.1.3. Stop the configuration process

The method family configuration process is stopped from two points: from the *Define Method Requirements* intention and from the *Construct an Application Method* intention depending on the manner to use the methodological knowledge for executing a process (or a set of activities) for which the method family is elaborated.

In the first case, the engineer does not realize the configuration of the method family as he knows well the given method field and can attain the main intention by using its expertise or he uses a tool available for this field.

- **By expertise.** The *By expertise* strategy is applied when an actor does not need an additional methodological assistance and uses his expertise and experience for executing processes related to the given method family.
- **Tool-based strategy.** This strategy is used when the engineer has a tool allowing to execute the processes of the given methodological field.

In the second case, the engineer uses the configured application method in order to execute the given activities. He can do it by applying the obtained application method or by integrating it with the existing methodologies.

- **By application.** Using this strategy, the engineer directly applies the obtained method without integrating it with an existing methodology.
- **By integration.** The *By integration* strategy is used when the engineer wants to integrate the configured application method with an IS engineering process. For this purpose, he can use one of the SME approaches allowing the method components integration [Ralyté *et al.*, 2003], for instance, the *By assembly* and *By extension* techniques.

4.2.2. Illustrative Example

We use an example from the RE field dealing with the elicitation of use cases in order to illustrate different possibilities of the method family configuration (Step by step, By sub-set selection, and By complete application method selection strategies).

The example is the *Crews-L'écritoire* approach described in [Ralyté *et al.*, 1999]. The corresponding map is given at Figure III.4.3.

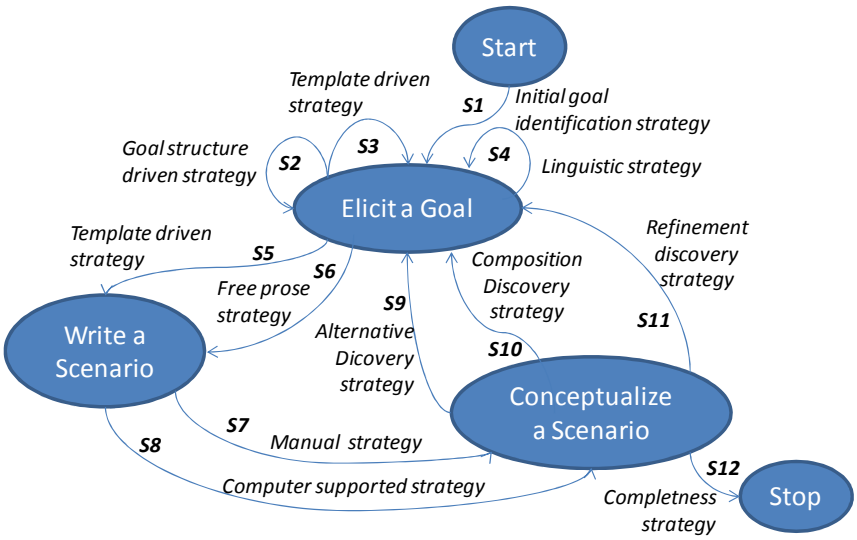


Figure III.4.3. The Crews-L'écritoire Method (the CL Map).

The *L'Ecritoire* method provides guidelines to discover functional system requirements expressed as goals and to conceptualize these requirements as scenarios describing how the system satisfies the achievement of these goals. This map contains three main intentions, namely 'Elicit a Goal', 'Write a Scenario' and 'Conceptualize a Scenario'. In this example, we have selected six indicators for carrying out the configuration: *Expert role*, *Expertise degree*, *Formality degree*, *Complexity degree*, *Relationships degree* and *Goal achievement degree*. Table III.4.1. illustrates these indicators applied to the method components.

Table III.4.1. Indicator's Values.

Method Components	Expert role	Expertise degree	Formality degree	Goal Achievement degree	Relationship degree	Complexity degree	Duration
S1	Analyst	1					15 mn
S2	Analyst	3	3	Low			10 mn
S3	Analyst	2	3	Low			10 mn
S4	Analyst		2	Low			15 mn
S5	Analyst	3	3	High			10 mn
S6	Analyst	1	1	High			5 mn
S7	Analyst	2	1	High			10 mn
S8	Analyst	1	3	High			15 mn
S9	Analyst			High	Medium	Medium	20 mn
S10	Analyst			High	Medium	Medium	20 mn
S11	Analyst			High	Medium	Medium	20 mn
S12	Analyst		1	High			5 mn

All components have all the same value for the *Expert Role* ('Analyst'). Two indicators (*Expertise degree* and *Formality degree*) have quantitative values between 1 and 3. For instance, if the value of *Expertise degree* is 1, it means that the engineer does not have to be an expert to use this component (as for *in free prose*). On the contrary, to use the Goal structure driven strategy will requires a high level of expertise as the goal has to be formalized in details, so the value is 3. The *Relationship degree* and *Complexity degree* indicators have a qualitative value equal either at Low, Medium or High. The *Duration* values are expressed in minutes and are impacted by the *Expertise degree* value. For instance, the component S1 will be realized in 15 minutes if the *Expertise degree* is equal to 1. If this indicator has a bigger value, the length of time used to realize the section will be shorter.

Thus, these values are defined at the contextualization step. The next step is to use these values for the configuration. In the following sub-sections, we illustrate three main configuration techniques applied to the Crews-L'écritoire method. Each technique is illustrated with one and multiple indicators used for configuration.

4.2.3. Step-by-Step Configuration

The usual way to use a map is to dynamically choose the method components to execute one by one.

Step-by-Step Configuration with Single Indicator. The engineer may use a specific indicator particularly relevant to its project, to help him to make his choice. First step is to select an indicator; the second is to determine the preference rule about it.

For instance, the navigation through the Map leads the engineer to execute the component S3. Five possibilities are offered to the engineer to continue the navigation. He may execute one of the three components which have the same target intention (S2, S3, S4) or he may go further in the Map to the intention *Write a Scenario* with the execution of the components S5 or S6. If its predefined indicator is the *Duration* with a preference to a lowest value, he will choose the component S6 which has the lowest value.

Step-by-Step Configuration with Multiple Indicators. The engineer may also use an aggregated value of several indicators (aggregation of the alternatives evaluations into a unique indicator, following a specific MC method). In this guidance, the first step is to select several indicators; the second is to aggregate values; and the third is to determine the preference rule about the results.

Within the preceding example, the navigation has led the engineer to execute the component S3. He can now execute S2, S3, S4, S5 and S6. He chooses three indicators: *Duration, Formality degree* and *Expertise degree* with a preference to a lowest final value. The normalized and aggregated values are described in Table III.4.2. Between these five components, the one that has the lowest value is the component S6.

Table III.4.2. Normalized values of Duration, Expertise degree and Formality degree.

Indicator	**S2**	**S3**	**S4**	**S5**	**S6**
Duration	10	10	15	10	5
Formality degree	3	3	1	3	1
Expertise degree	3	2	1	3	1
Normalized values					
Duration degree	0,5	0,75	0,75	0,5	0,25
Formality degree	1	1	0,33	1	0,33
Expertise degree	1	0,67	0,33	1	0,33
Aggregated Value	***2,5***	***2,42***	***1,41***	***2,5***	***0,91***

4.2.4. By Sub-set Selection

This strategy is applied when the goal is to eliminate components, which are not appropriated for the given indicators values.

Components Sub-set Selection with Single Indicator. It is applied when only one indicator is available. The engineer has to define the needed value of this indicator in order to select an adapted sub-set of components.

If, for instance, the engineer is a beginner, the *Expertise Degree* indicator must have a value equal to 1 (low). Using this approach allows to delete all the components which are too difficult for the user. The components sub-set of the CL Map is then composed of the following components: S1, S4, S6, S8, S9, S10, S11, and S12. An engineer with a low level of expertise may execute all these components.

Components Sub-set with Multiple Indicators. This guidance has the same goal as the previous one but uses an aggregated value of several indicators following an MC method. Several functions may be applied (sum, maximum, minimum, average, weighted sum).

For instance, the engineer selects the *Expertise degree* and the *Formality degree* as the two more important indicators and he wants to minimize their values. As aggregation rule, the engineer chooses to calculate the average value. The indicators do not have a value for all the components. In order to obtain an aggregated value, we consider that a null value is equal to the lowest value of the indicator (see italic values in Table III.4.3.). For instance, if the value of the *Expertise Degree* is not specified for a component, we assume that any engineer may execute it and we affect the value 1 to this indicator. Once the aggregated values have been defined, the engineer defines the maximal average value that he could accept (here 1,5). The final components sub-set includes: S1, S4, S6, S7, S9, S10, S11, and S12.

Table III.4.3. Aggregated Value of Expertise degree and Formality degree of the CL Map.

Indicator	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**S8**	**S9**	**S10**	**S11**	**S12**
Formality degree	*1*	3	3	*1*	3	1	1	3	*1*	*1*	*1*	1
Expertise degree	1	3	2	*1*	3	1	2	1	*1*	*1*	*1*	*1*
Aggregated value	***1***	***3***	***2,5***	***1***	***3***	***1***	***1,5***	***2***	***1***	***1***	***1***	***1***

It is interesting to note that the only difference between this sub-set and the sub-set from the previous section (4.2.1) is the component S8 replaced by the component S7 between the intentions *Write a Scenario* and *Conceptualize a Scenario*. The adjunction of the *Formality degree* to the *Expertise degree* has only this little impact.

4.2.5. By Complete Application Method Selection

In order to apply this configuration technique, we suggest using the graph theory. We foresee enhancing the maps guidance on the operational level by their expression in terms of graphs. This offers the process execution associated to the map with the control of the map navigation.

In this section, we show how a method family modelled with map could be represented as a graph and illustrate the usage of the graph algorithms for configuring method families.

4.2.5.1. Representing Method Families by Applying the Graph Theory

Our aim is to enhance the guidance mechanisms of the map execution by reusing graph algorithms. The Graph theory offers a lot of techniques to be used on graphs. For instance, the shortest path problem is the problem of finding a path between two vertices such that the sum of the weights of its constituent edges is minimized. On a valuated map, it may be useful to find a path through the map which will satisfy some requirements of the IS engineer (on time, cost and so on). The execution of the map is then more flexible as the engineer has the possibility to change the weight values of the map sections following the project at hand.

There is confusion between maps and graphs. Even if maps are visually constructed as graphs, their use is completely different. These differences are semantically based on the two different levels used: the intentional level of the MAP model and the operational level of the graphs. Consequently, our aim is to propose a possible mapping between maps and graphs in order to use all the graphs techniques already defined in the literature.

Graphs

The graph theory was born to study problems such as how to visit some places only once on a walk [Euler, 1736]. In mathematics and computer science, graph theory is the study of graphs: mathematical structures used to model pairwise relations between objects from a certain collection.

Figure III.4.4. shows the graph structure model.

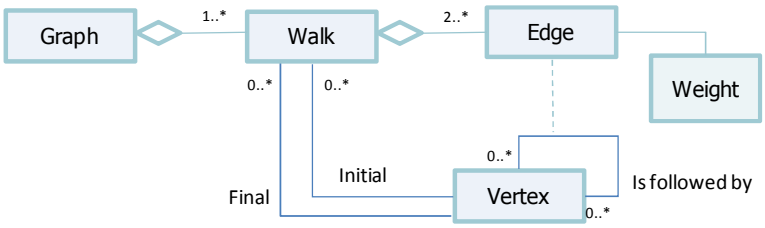
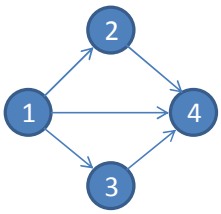


Figure III.4.4. Meta Model of Graph.

Graphs are represented graphically by drawing a dot for every vertex, and drawing an arc between two vertices if they are connected by an edge. If the graph is directed, the direction is indicated by drawing an arrow, as shown at Figure III.4.5.

A graph can be represented as *G=(V,E)*, where *V* and *E* are disjoint finite sets. We call *V* the *vertex set* and *E* the *edge set* of *G* [Berge, 1985]. A *walk* is an alternating sequence of vertices and edges. An example of a walk is given on Figure III.4.5. as this graph offers three different walks to go from the vertex 1 to vertex 4 (either directly or going through vertex 2 or through vertex 3). There is a distinction in graph theory between a *path* and a *walk* as a path is a walk with no repeated vertices. A walk begins with an initial vertex and ends with a final vertex.



V={1,2,3,4}
E={(1,2), (1,3), (1,4), (2,4),(3,4)}
$W_{1\text{-}4}$={(1,4), (1,2-2,4), (1,3-3,4)}

Figure III.4.5. Graph Example.

A multigraph or pseudograph is a graph which is permitted to have multiple edges, (also called “parallel edges”), that is, edges that have the same end nodes. Thus, two vertices may be connected by more than one edge. Cycles are allowed in these graphs. A cycle is a path which ends at the vertex where it begins.

A weighted graph associates a label (weight) to every edge in the graph.

MAP and Graph correspondence

The transformation from map to graph is shown at Figure III.4.6.

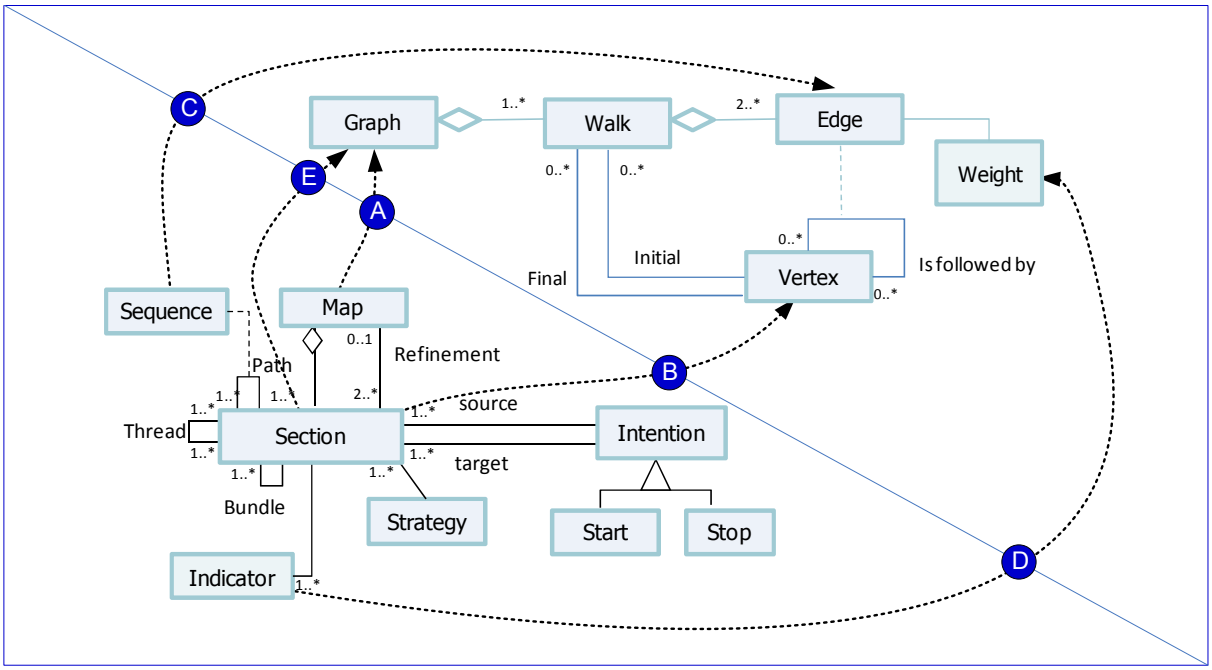


Figure III.4.6. MAP and Graph Correspondence.

Each map corresponds to a graph (A). Each section of the map is transformed into a graph vertex (B) and each identified section sequence is shown as an edge of the graph (C).

We may also identify the correspondence between the MAP indicator and the graph weight (D). The section weights are evaluated as indicators that can be applied on the graph edges. The valuations of the graph edges could be obtained based on the aggregated values of sections: the section weight becomes the same valuation on all entering edges for a given section on the graph. We assume that each section has the same value independently of the previously realized section.

The MAP model allows refining a section with another map. This abstraction level is kept with the graphs as a node may also be refined as another graph.

These correspondences are summed up in Table III.4.4.

Table III.4.5. Correspondences between Maps and Graphs.

Map concept	Graph concept	Correspondance
Map	Graph	Each map is represented as a specific graph.
Section	Vertex	Each section of the map is represented as a vertex in a graph.
Sequence	Edge	The concept allowing the navigation in a map is the sequence of sections in the map, which may be represented as edges in a graph.
Indicator	Weight	The guidance parameters called indicators in the map are the weights in a graph.
Path	Walk	The set of section sequences (path) is a graph walk.
Thread	Set of adjacent vertices	The thread is the possibility to attain a target intention from a source intention with several strategies. In a graph, the representation of this set of sections will be the set of vertices having two specific edges, the one shares the same start-vertex and the other the same end-vertex.
Bundle	Set of adjacent vertices	The thread is the possibility to attain a target intention from a source intention with several strategies but with an exclusive OR which means that only one of these sections may be used in the complete navigation. The representation on the graph is the same than for the thread.

Once a map is expressed through a graph, the algorithms of the graph theory could be applied for the configuration.

4.2.5.2. Application of the Graph Theory Algorithms in Configuration

This approach allows finding an optimal path through the method family using the graph theory algorithms.

Graph Theory Algorithms by Single Indicator. Possible algorithms to apply are the *Shortest path*, the *Hamiltonian path* (path navigating through all the sections of the Map) and so on. The engineer selects an indicator, identifies all possible paths, and calculates values for each path (for quantitative indicators this is a sum and for qualitative indicators it can be a given value from the predefined set as [ENUM]).

For instance, the engineer decides to select a path, which minimizes time for executing the CL Map. He applies the graph theory algorithm for searching the Shortest path following the indicator *Duration*. The *Duration* values for all components of the map are indicated in Table III.4.1. Recall that in this example, the *Expertise degree* indicator impacts the *Duration* (an engineer with a low level of expertise will execute the component in more time than if he was an expert). The *Duration* values are given in the case when the *Expertise degree* is low. The possible paths set may be quite big as the Map contains cycles, iterations and back paths. To simplify our example, only the four shortest paths (which do not contain any cycle or iteration) are retained:

- Path 1: {S1, S5, S7, S12} = 15 mn+10 mn +10 mn +5 mn = 40 mn;
- Path 2: {S1, S5, S8, S12} = 15 mn +10 mn +15 mn +5 mn = 45 mn;
- Path 3: {S1, S6, S7, S12} = 15 mn +5 mn +10 mn +5 mn = 35 mn;
- Path 4: {S1, S6, S8, S12} = 15 mn +5 mn +15 mn +5 mn = 40 mn.

The shortest path is the Path 3 as it is the one which will be the quicker to execute.

Graph theory algorithms by Multiple Indicators. The Graph theory algorithms may also be used with multiple indicators. Instead of using only one of them to select the better path, several indicators values are aggregated together in order to obtain a single value for each section. Then, these values are used to select the optimal Map path like in the previous guidance. Several functions may be applied (sum, maximum, minimum, average, weighted sum) for aggregating indicators values.

In our example, the engineer selects three indicators: *Duration*, *Formality degree*, and *Expertise degree*, which are quantitative and have different measure scales. In order to obtain compatible scales for comparing them, normalization must be applied. In this case, the normalization is calculated as a percentage of the maximal value for each indicator. For this, the *Duration* values [5; 20], *Formality degree*, and *Expertise degree* values (ENUM:1, 2, 3) are reduced to the same scale [0; 1]. The normalized values are presented in the Table III.4.5.

Table III.4.5. Normalized values of Duration, Expertise degree and Formality degree.

Indicator	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**S8**	**S9**	**S10**	**S11**	**S12**
Duration	15	10	10	15	10	5	10	15	20	20	20	5
Formality degree	1	3	3	1	3	1	1	3	1	1	1	1
Expertise degree	1	3	2	1	3	1	2	1	1	1	1	1
Normalized values												
Duration degree	0,75	0,5	0,75	0,75	0,5	0,25	0,5	0,75	1	1	1	0,25
Formality degree	0,33	1	1	0,33	1	0,33	0,33	1	0,33	0,33	0,33	0,33
Expertise degree	0,33	1	0,67	0,33	1	0,33	0,67	0,33	0,33	0,33	0,33	0,33
Aggregated Value	***1,41***	***2,5***	***2,42***	***1,41***	***2,5***	***0,91***	***1,5***	***2,08***	***1,66***	***1,66***	***1,66***	***0,91***

The engineer wants to minimize Duration, formality and expertise degrees, so he chooses the shortest path. The function used on the aggregated values is the sum.

- Path 1: {S1, S5, S7, S12} = 1,41 + 2,5 + 1,5 + 0,91 = 6,32
- Path 2: {S1, S5, S8, S12} = 1,41 + 2,5 + 2,08 + 0,91 = 6,90
- Path 3: {S1, S6, S7, S12} = 1,41 + 0,91 + 1,5 + 0,91 = 4,73
- Path 4: {S1, S6, S8, S12} = 1,41 + 0,91 + 2,08 + 0,91 = 5,31

The shortest path taking into account the three selected indicators is the Path 3.

An example of the map expression with graphs describing the construction process of an O* model is given in [Deneckère *et al* , 2009a].

4.3. Configuration of the DM Method Family

In this section, we explain the usage of the configuration process in the case of the DM method family. The expression of the configuration process applied to the DM method family represents a sub-map of the Method family configuration map illustrated at Figure III.4.1. We have selected the sections applicable to the DM method family. The corresponding map is shown at Figure III.4.7.

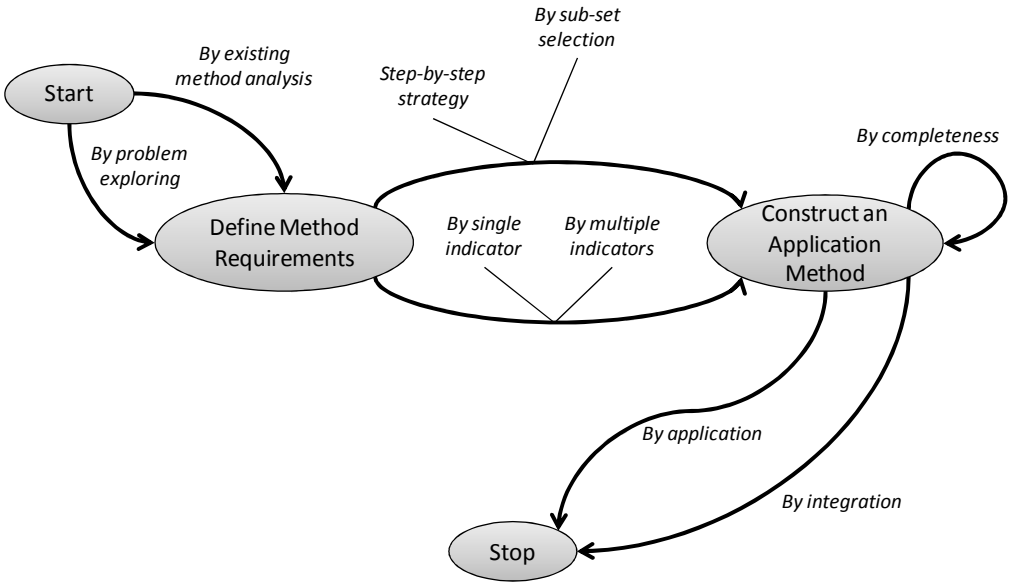


Figure III.4.7. The Method Family Configuration Map for the DM Method Family.

4.3.1. Define Method Requirements

The *Define Method Requirements* intention includes the identification of the decision object, decision problem, intention and situation. The *DM object* is an artifact being the subject of decision-making. The *intention* means the goal that the actor has in the DM process. The most used goal is to make decision. This goal can be completed by other ones: take into account multiple viewpoints, consider different aspects for comparing alternatives, arbitrate between aspects, and so on. The *situation* means the available elements that can be used in the DM process. For instance, it can be a set of alternatives, a mandatory for a given case set of criteria, alternatives evaluations, and so on.

The definition of the DM method requirements could be done by two ways:

- By problem exploring;
- By existing method analysis.

4.3.1.1. By problem exploring

The engineer begins from zero when he does not have any methodological guideline for providing him with the aid for decision-making. The actor must resolve a problem and often does not have alternative solutions. This strategy allows guidelines for identifying different elements necessary for DM.

The guideline of this strategy contains the following steps:

1. Formalize the DM object; assign a type (product or process)
2. Select a problem from the list (choice, sorting, ranking, or description)
3. Select intentions from the list
4. Select situation's elements from the list

> ***Example:*** For instance, an engineer wants to prioritize requirements that he has in this project. By applying this strategy, he defines:
>
> ```
> DMObject.name: = Requirement
> DMObject.type: = Product
> Problem.type: = Ranking
> Intention: = Make decision, Define criteria, Define relative
> importance of criteria
> Situation: = DM Object defined, Alternatives defined
> ```

4.3.1.2. By existing method analysis

In this case, a method exists but must be improved. Therefore, several DM elements are already defined. This strategy allows analyzing existing DM elements and to complete an existing method.

This strategy is carried out as follows:

1. Analyze the existing method;
2. Identify the DM object already specified in the method; Select situation's elements from the list;
3. Select intentions from the list.

> ***Example:*** For illustrating this strategy, the example of the cost-value approach could be mentioned (See Section 2.4.1. of this Part for the details of this example). The engineer uses this method for prioritizing requirements. However, he considers that the alternative list is too long to be able to apply the AHP method used in the cost-value approach. Thus, he wants to decrease the number of alternatives available for prioritization. In this case, he will select the intention *Refine alternatives list* in order to find the appropriate DM component.

4.3.2. Construct an Application Method

The *Construct an Application Method* intention aims at selecting one or more DM components available in a given case and to unify them within a new method. The components selection is based on information defined at the *Define Method Requirements* intention.

- Step-by-step strategy;
- By sub-set selection;
- By completeness.

Both *Step-by-step* and *By sub-set selection* strategies could be applied using the single and multiple indicators.

4.3.2.1. Step-by-step strategy

An engineer navigates through the map representing the DM method family by selecting a component to execute and applying it. At each stage, an engineer can select between intentions to attain, strategies to realize, or activities to perform.

> ***Example:*** For instance, we return to the case of the engineer who wants to prioritize requirements (See Section 4.3.1.1.). Once his requirements are identified, he is positioned on the Define alternatives intention as he already has his requirements list. He must select the following step to follow. He can choose between three intentions: Define alternatives, Define criteria, and Evaluate alternatives.

4.3.2.2. By sub-set selection

Different DM method components are selected based on the engineer's requirements identified on the previous intention.

> ***Example:*** For instance, the junior engineer uses the DM method family. Thus, the available expertise degree is low which corresponds to the value 1 in the DM method components characterization. In this case, the selected DM method components are S3, S4, S5, S6, S7, S8, S11, S12, S13, S14, and S15 (See Figure III.4.8.).
>
> As we can see, the obtained method cannot be applied. The engineer must add other criteria for the components selection, or apply other strategies (for instance, by completeness) in order to construct a complete application method.

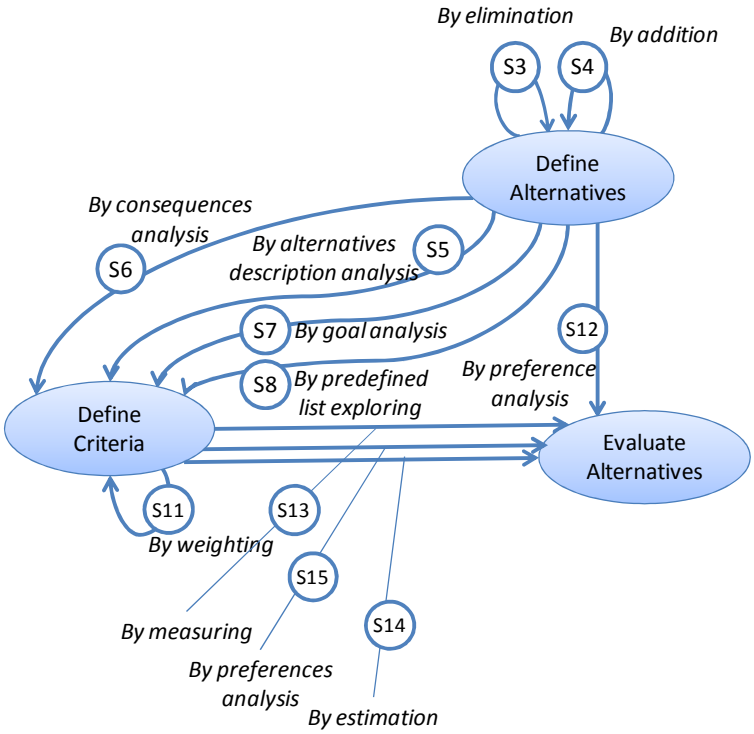


Figure III.4.8. DM Method Components for Illustrating the By Sub-Set Strategy.

4.3.2.3. By completeness

The identified *by sub-set selection* DM method components are checked with regards to their completeness.

> ***Example:*** For instance, the first set of DM method components includes the method components S1, S15, S16, S21, and S24. Firstly, these components are positioned on the DM method family map (marked by the solid lines at Figure III.4.9.).
>
> This shows that these components must be completed by those that allow to define criteria. Secondly, the four potential DM method components are identified in order to provide a complete DM application method (S5, S6, S7, and S8 marked by the dashed lines at Figure III.4.9.). Thirdly, only one component must be selected between them in order to finalize the application method, for instance, the component S7.

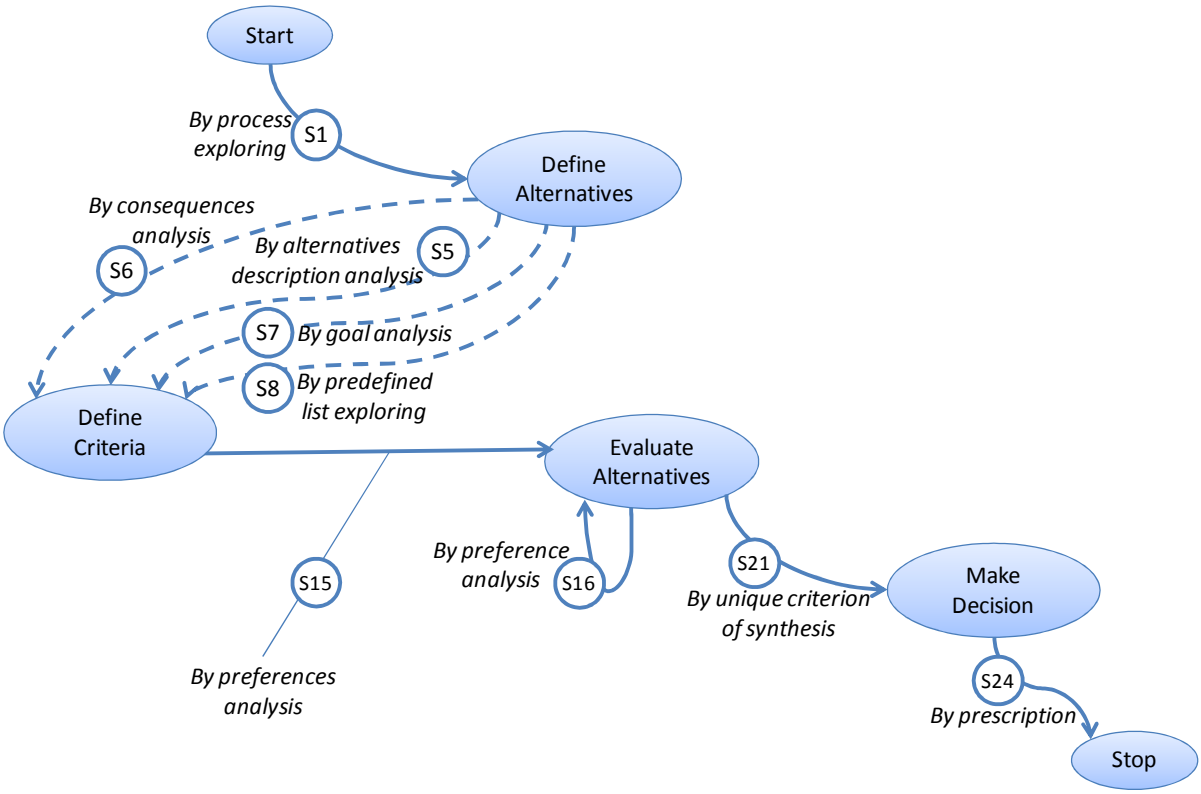


Figure III.4.9. Application Method for Illustrating the Completeness Strategy.

4.3.3. Stop the configuration process

Once the application method is constructed, it can be applied directly or integrated with the exiting method. Both strategies use the techniques from the method engineering field.

4.4. Conclusion

We have presented in this Chapter a process to configure method families and its application to the case of the DM method family. This process offers different guidelines for configuring a method family. Each time, when this process is applied, an application method is derived from the method family.

The configuration process is used in order to enhance an existing method from the same field, or to create a new method. In both cases, the configuration process takes into account actors requirements and specific context characteristics in order to construct an application method adapted to this context.

The generic method family configuration process is not completely used in the case of the DM method family. In fact, the *By complete application method selection* strategy is not suitable for the DM method family as its map contains multiple loops and the calculation of all possible method lines (such as paths

using the graph theory) is too complicated. However, the given strategy could be used for other cases like the CREWS-L'écritoire one.

In whole, the configuration process completes the DM method family by providing guidelines for using the DM map.

Part IV. MADISE in Practice

This Part contains two Chapters. We present the validation of the MADISE approach in Chapter 1. We conclude our thesis in Chapter 2.

Chapter 1: Validation of the MADISE Approach

1.1. Introduction

In this Chapter, we validate our proposal by two ways: by application and by comparison. In the first case, we apply the MADISE approach to make decisions in the Business Process Management (BPM). In the second case, we compare the proposed approach with five known methods of prioritizing requirements by using the completeness and flexibility measures. Before describing these two cases, we briefly introduce the tool supporting the MADISE approach which is a repository of the DM method components.

1.2. MADISE Tool

The goal of the MADISE Methodological Repository (MADISE Repository) is to provide the methodological support for realizing DM activities. From this point of view, the MADISE Repository aims at storing the DM method components and at supporting the DM method family configuration process.

The MADISE Repository is developed as a Web site and contains the following elements:

- MADISE DM Map General Description
- Hierarchy of DM Method Components
- Theoretical Background
    - DM and DM Methods

- Method Engineering and Method Components
- Modeling Approach (Map)

- Configuration of the DM method family (under construction)

The database structure is presented at Figure IV.1.1.

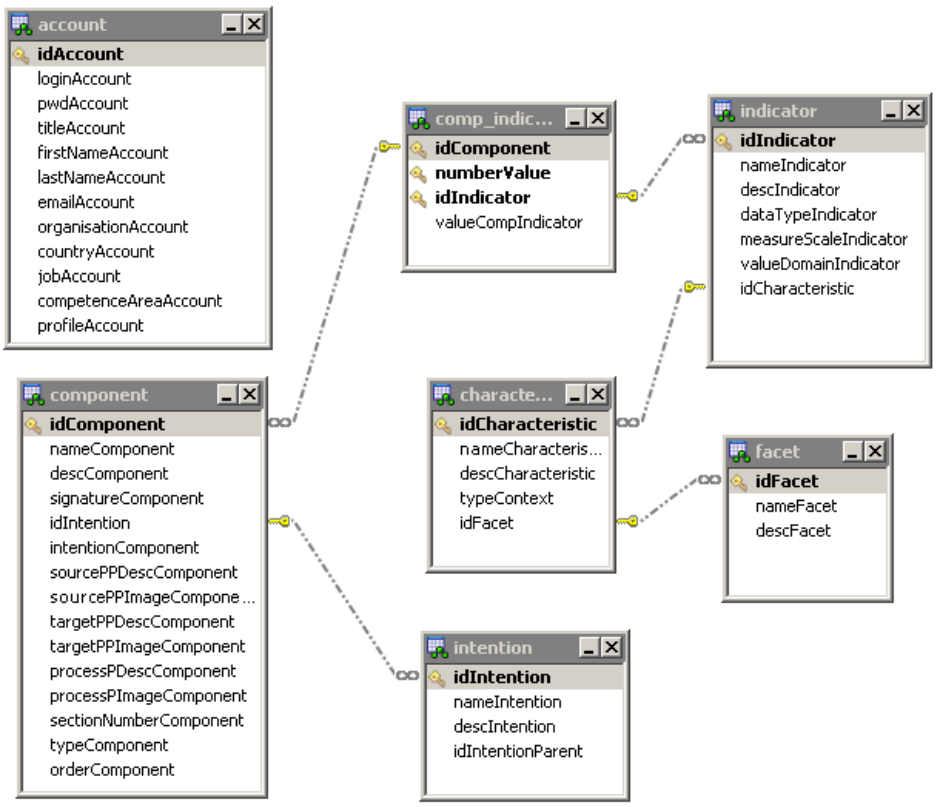


Figure IV.1.1. Database Structure.

The database contains all necessary elements for displaying DM method components. Information about them is registered in table *component*. This table is related to the table intention and to a table which allows to specify the context of the DM method components. Each component is characterized by a set of indicators which can have one or more value for a given component. Indicators are described in table *indicator*. Indicators are related to the corresponding context characteristics organized into facets.

In addition, the database contains the table *account* allowing to connect to the DM method components repository.

The MADISE Repository functioning is given at Figure IV.1.2.

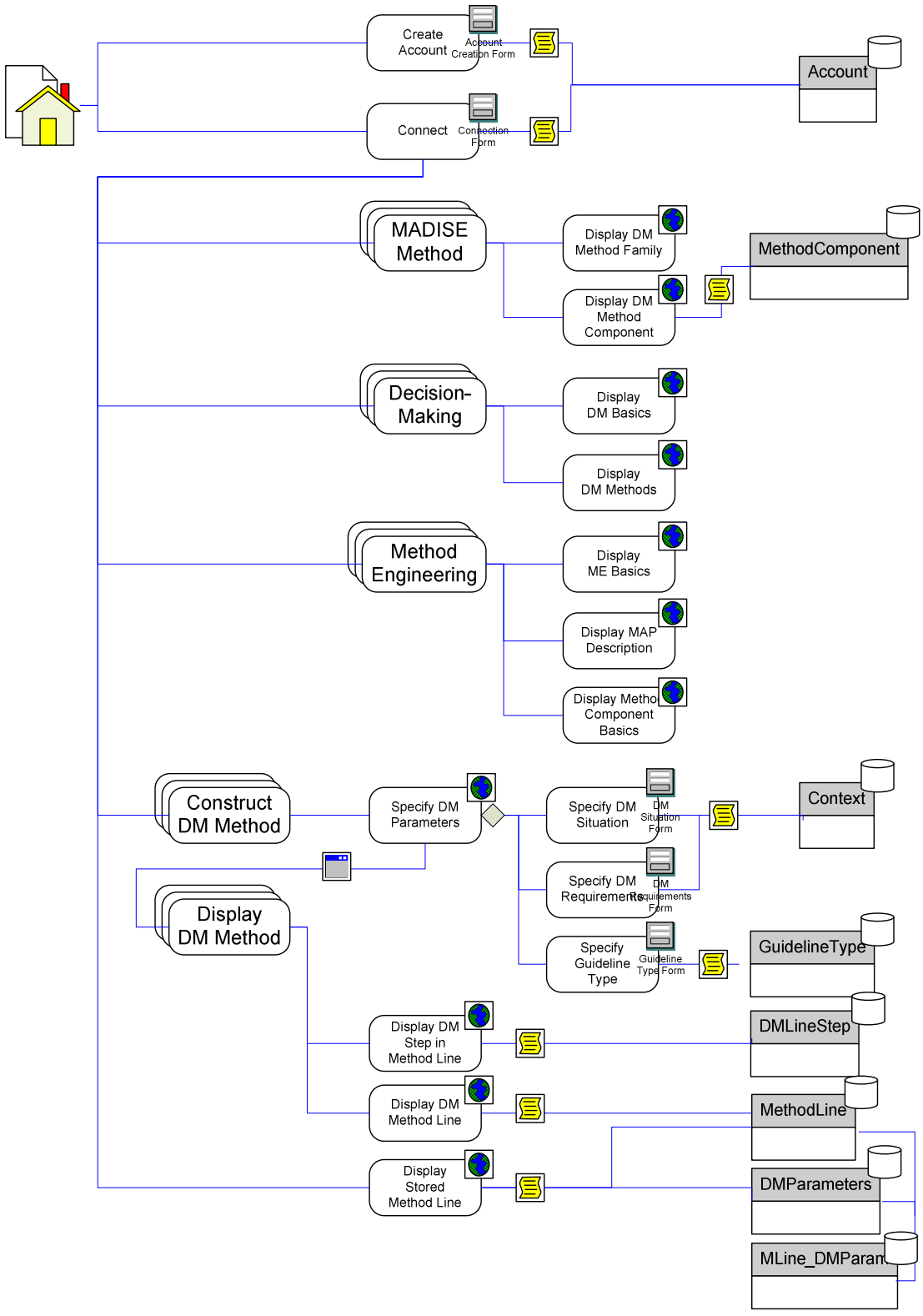


Figure IV.1.2. MADISE Repository Functioning.

The first step is to connect to the MADISE Repository. Once connected, an engineer can consult the DM method family, the DM method components, and the theoretical background of the MADISE approach.

Figure IV.1.3. shows several screenshots of the MADISE repository, namely, the DM method family, a sample of a DM method component, and DM methods.

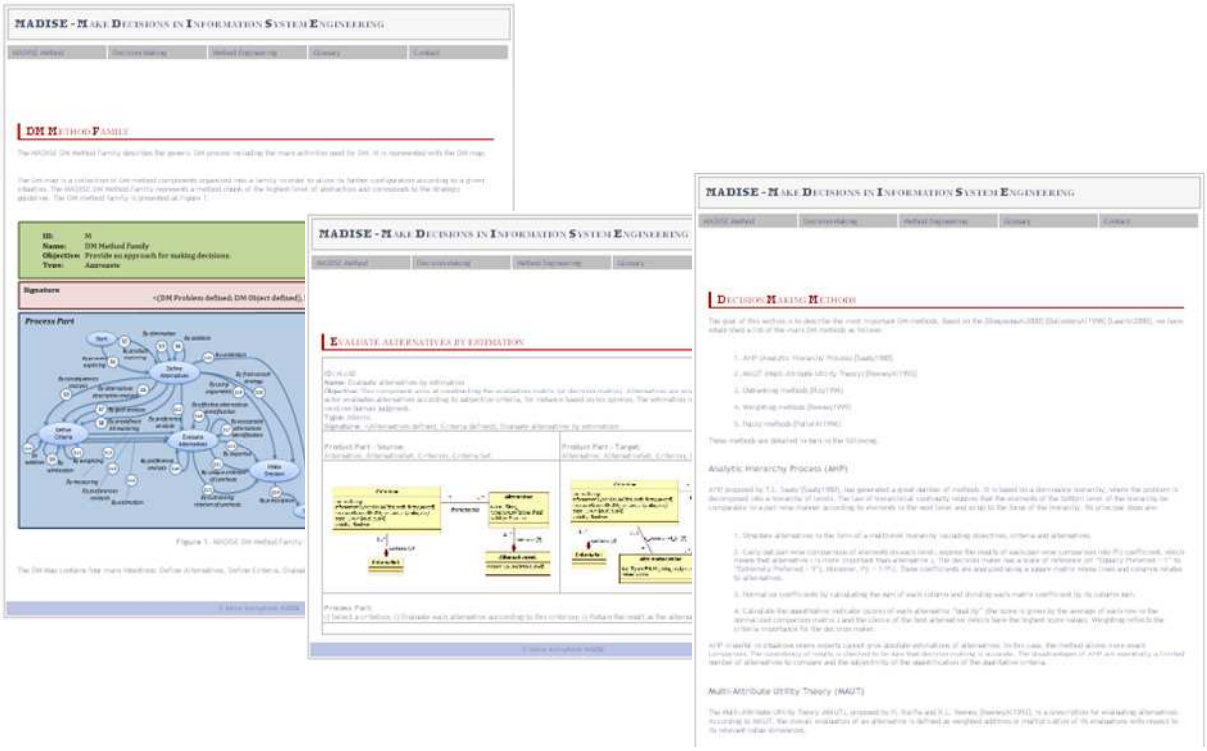


Figure IV.1.3. MADISE Repository Screenshots.

The MADISE repository could be accessed on the page http://elena.kornyshova.free.fr/.

1.3. Business Process Prioritization

In order to validate our approach, we have selected the case of Business Process Management (BPM). By developing this case, we will demonstrate the following hypothesis.

> ***Hypothesis:*** *Usage of the DM method family will allow a better definition of the key (priority) business processes than the existing BP prioritization methods by providing the complete DM guidelines adapted to a given case.*

In this section, we analyze the existing DM methods in this field, we configure the DM method family for prioritizing business processes (BP), and we apply the obtained method in a concrete case of a company of the electronics industry domain.

1.3.1. Decision-Making in Business Process Management

All business processes provide achievement of general purposes of organization. However, each process is intended to satisfy certain requirements of an appropriate group of stakeholders. An organization must always estimate the contribution of each process to achieve strategic goals and determine major processes. These processes require more attention and more resources. They bring the decisive contribution to an actual and future situation of organization. These processes are called "key business processes" (KBP) [Sachdeva *et al.*, 2005]. BPs prioritization is used by companies to define the most important development axes, increase the reaction speed to environment changes, optimize the expenditure, and consequently, improve competitiveness.

The KBP identification may improve the decision-making in many areas. The examples are:

1. Selection of business processes for improvement;
2. Choice of processes corresponding to organization strategy;
3. Definition of R&D main lines;
4. Elimination of useless processes as much as possible;
5. Choice of the first business processes that need to be secured.

The KBP identification allows more flexibility and adaptability by having quicker reactions to strategic environment changes. In fact, the business process prioritization has several advantages: concentration on the most expensive processes allowing the cost optimization, best knowledge and satisfaction of stakeholders' interests, often contradictory and, generally, competitiveness improvement. These effects can be multiplied by quickness and dynamic adaptation.

Dealing with Key BPs supposes that decision makers know business processes priorities or are able to define them, at least intuitively. The intuitive approach is viable when there is a limited number of BP. However in most cases, managers face large scale problems with a large number of BPs, large size BPs, variability, and often different versions of BPs through time. The combination of these issues leads to a combinatory explosion of the number of artifacts to deal with, hence the need for a better prioritization support.

In the following, we present an overview of existing business processes prioritization approaches. After a brief description of these approaches, we compare them and give some conclusions about their pros and cons.

1.3.1.2. Summary of Existing BP Prioritization Approaches

There are only a small number of decision-making methods that propose to guide the BP selection. Four approaches are particularly considered in our review: (i) Hammer and Champy's, (ii) Robson and Ullah's, (iii) PROSCI, and (iv) Mazur et al.'s approach.

Hammer and Champy [Hammer *et al.*, 1993] propose to analyze all BP under three different perspectives in order to select those that need reengineering. These are problems, importance and

feasibility. First, all processes for which a problem can be identified are chosen. Then, the importance of these problems for the organization is analyzed. Last, a feasibility control is carried out in order to verify if expected results will cover related expenses. It can be seen that this approach is informal and is not relevant when there is a large number of BPs.

Robson and Ullah [Robson *et al.*, 1996] propose a methodology to sort BPs for reengineering. In this approach, BPs are analyzed in relation with critical success factors (CSF). First, relevant CSF are listed, then each BP is estimated along a five-grade scale according to all CSF. A weighted sum is generated for each process; weights represent relative importance of CSF. In this approach, value is considered as a complex concept that concerns all BPs of the organization. Besides, the authors suggest to analyze BP functioning (from very good to bad, according to five-grade scale). Three levels of priorities are finally considered: reengineering, improvement and supervision. BP that contributes to many CSF and have bad functioning are considered as potential for BP reengineering.

PROSCI [Crowe *et al.*, 1997] uses a BP taxonomy to identify reengineering opportunities. The authors suggest that relations exist between strategic objectives and BPs. The first step of this approach consists in establishing a taxonomy of BPs. In the second step, the relations between strategic objectives and BPs must be elicited. To achieve this, an influence diagram is drawn using decision trees where every BP is embedded in a main decision node, strategic objectives are drawn as a chance node, and main decision nodes are linked to chance nodes. Thus, the influence of each BP on every strategic objective is taken into account. Relationships are in the form of probability distributions, which reflect the stochastic nature of influences that business processes have on the strategic objectives. The final BP evaluations are obtained by weighted sums of chance nodes in which weights are assigned to chance nodes depending upon their order of importance.

Mazur et al. [Mazur *et al.*, 2000] propose an approach to BP selection based on the weighed sum. In this approach, the calculation is made according to pre-defined criteria: influence on customer, variability, functioning, and importance for business. Each BP is measured towards all criteria with five-grade scale, and then the weighted sum is calculated.

A comparison of these approaches is present in the next sub-section.

1.3.1.3. Comparison of BP Prioritization Approaches

As Table IV.1.1. shows, the four selected approaches were compared along two dimensions. The first dimension deals with the criteria used by the approaches to compare BP. The second dimension concerns the rules proposed to carry out BP selection.

Table IV.1.1. Comparison of BP Prioritization Approaches.

BP Prioritization Approach	***Criteria***	***Selection Rules***
Hammer and Champy	Problems, importance and feasibility of BP	Not specified
Robson and Ullah	Critical success factors	Quantification with 5-grade scale, weighted sum
PROSCI	Strategic objectives	Decision tree, weighted sum
Mazur et al.	Importance for customers and business, variability, functioning	Quantification with 5-grade scale, weighted sum

Several remarks can be drawn from this table:

(i) there is only a limited set of criteria to support BP comparison;
(ii) most criteria are abstract (e.g. problems, or importance for customers and business), and the authors do not show how these criteria relate to actual BP performance indicators; and
(iii) there are only two kinds of selection rules: weighted sum and bi-dimensional surface. The drawback of weighted sum is that it requires homogeneous criteria. On the other hand, bi-dimensional surface have the disadvantage of limiting the maximal number of criteria.

In order to avoid these issues, we suggest using the DM method family for the BP prioritization.

1.3.2. Case Study General Description

This section presents a case study undertaken at a company in the electronics industry. The company Filkon operates on the market of electronic components. The company is the manufacturer of ceramic capacitors and EMI (electromagnetic interference) filters. Its consumers are major industrial companies of different military and civilian industries, small and medium businesses.

This company is in dire financial conditions: a low rate of profitability, delays in payments of obligations and credits, and the drop in orders. Overall, the company is in crisis, which resulted in the bankruptcy announcement. The question of its existence is directly linked with the ability to sort out the situation and take steps to resolve the crisis.

The purpose of the experiment was to choose among business processes those to be reengineered in the context of continuous improvement. More precisely, the management was interested in the processes that had the greatest influence on achieving strategic goals. As a result of this experience, the company expected to identify one or two processes which reengineering would bring maximal value with minimal drawbacks.

1.3.3. DM Method Family Configuration

The used configuration process in this case is given at Figure IV.1.4.

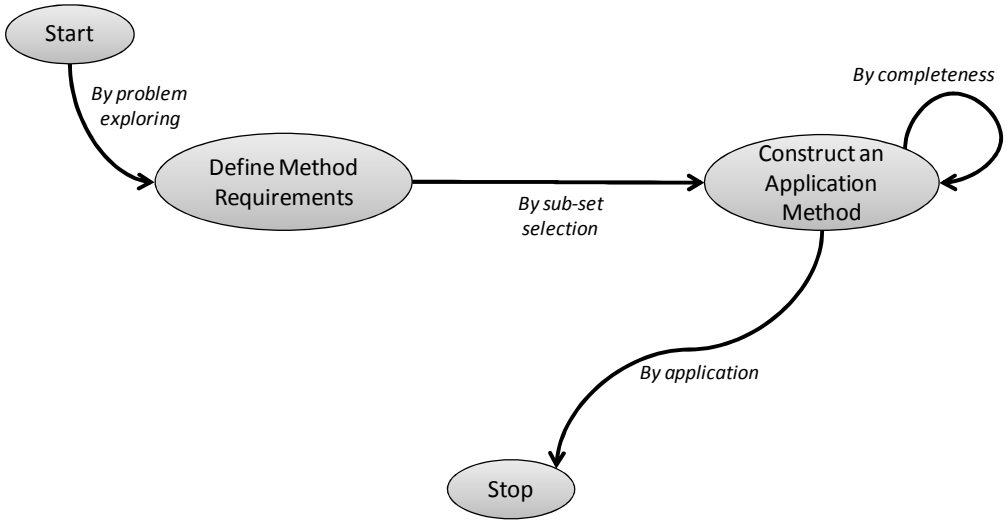


Figure IV.1.4. Configuration Process in the Case of BPM.

By problem exploring, the following elements are identified:

```
DMObject.name: = Business Process
DMObject.type: = Process
Problem.type: = Choice
Intention: = Make decision, Define criteria, Define relative
importance of criteria
Situation: = DM Object defined, Alternatives defined
```

The *set of potential business processes* has already been developed based on the "Process classification framework" proposed by APQC [Process Classification Framework, 1996]. The list of the business processes includes:

- BP1. Understand Markets and Customers;
- BP2. Design Products;
- BP3. Market and Sell;
- BP4. Produce and Deliver;
- BP5. Invoice and Service Customers;
- BP6. Develop and Manage Human Resources;
- BP7. Manage Information Resources;
- BP8. Manage Financial and Physical Resources.

By sub-set selection, we have pre-selected the following DM method components (See Table IV.1.2.).

Table IV1.2. DM Components Selected for the BPM Case Study.

Section	*DM Method Component*
S7	Define criteria by goal analysis
S8	Define criteria by predefined list exploring
S11	Define criteria weighting
S13	Evaluate alternatives by measuring
S14	Evaluate alternatives by estimation
S22	Make decision by outranking relation of synthesis
S24	Prescribe decision

The obtained DM application method is illustrated at Figure IV.1.5.

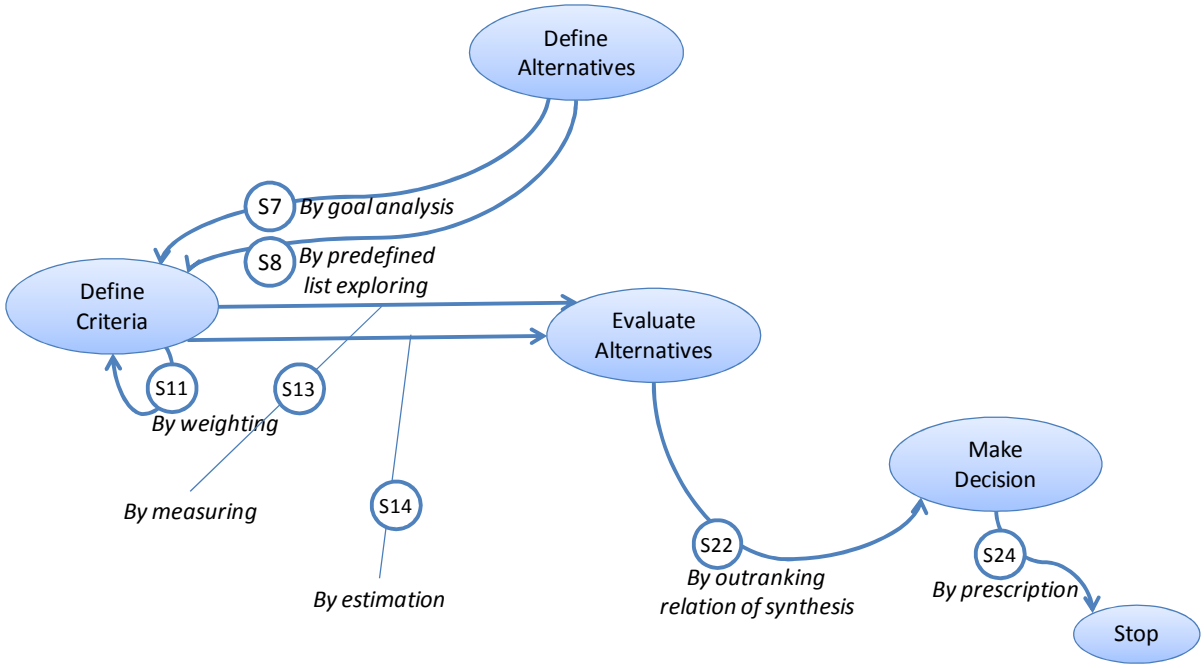


Figure IV.1.5. DM Application Method in the Case of BPM.

1.3.4. Application of the Obtained DM Method

In this section, we detail the application of obtained DM application method step by step.

1.3.4.1. Define criteria by goal analysis

The definition of the criteria based on the goal analysis requires the review of the strategic goals of the company. For this purpose, the analysis of the Balanced Scorecard (See Table IV.1.3.) is undertaken.

Table IV.1.3. Balanced Scorecard of the Company.

Objective	Indicator	Target Value
Financial performance		
Growth's accelerating	Increase of turnover	5%/year
	Increase of turnover for new customers	10%/year
	Increase of turnover for new products	10%/year
Customers		
Customers' satisfaction increasing	Increase of "in time" deliveries for customers	95% of total
	Organization image improving	9 points on 10
Customers' relationship developing	Number of partnership increasing	15/year
	Increasing of returning customers' share	15%/year
Market part increasing	Market part increasing for capacitors	5%/year
	Market part increasing for filters	10%/year
	Market part increasing for varistors	5%/year
Internal Processes		
Production quality improvement	Productiveness increasing	5%/year
	"Time-to-market" decreasing	6 months
	Increase of productive capacity use	10%
Supply improvement	"In time" supplies number stabilization	100%
Learning and growth		
Ensure personnel involvement	Number of learned persons	20% of personal/year
	Conditions of work satisfaction improvement	8 points on 10
IT development	Automation	2 business functions/year

Based on this data, the *BP contribution to strategic goals* is identified as criterion which presents the degree of influence of BPs on organizational performance. All goals are selected for studying the impact of BP. The preference rule associated to this criterion is maximal as business processes having the greatest impact on goals are more important for the company.

1.3.4.2. Define criteria by predefined list exploring

By predefined list exploring the following criteria are selected: cost, size, contribution to the problem resolution, life cycle step, and customer.

BP contribution to problems resolution means that BP reengineering (BPR) should help to resolve some decision problems. The problem at hand is to know which BP improvement would bring the greatest result. We defined improvement by the contribution of processes to problems that could be solved by process reengineering. To make the analysis closer to reality the frequency of occurrence

and threat degree were used as weights. For frequency, the scale was: 0 – never, 1 – sometimes, 2 – often, 3 – regular. For threat degree, the scale was estimated on a three-level grade: 1 – low, 2 – medium, 3 – high. This function aims at maximum.

BP costs are defined as the number of persons working on the BP. Preference rule is minimum.

BP sizes are defined by the quantity of sub-processes, which we believed would reflect their importance in the company. The preference function aims at maximum.

The purpose of *BP life cycle steps* is to define the administrative influences required for process reforming. The following steps are defined: creation, development, stable functioning, regress, and destruction. Indeed, it is found that process reengineering is needed or at least acceptable for processes in the state of development, regression or stable functioning. Reengineering is felt less preferable for BPs in state of creation, destruction and stable functioning. Therefore, the preference rule is defined as: (development = regress) ≥ stable functioning > (creation = destruction).

In the given company, *BP customers* could be external or internal. External processes add value for organization's customers, therefore they were considered as more important. The preference rule is: external ≥ internal.

In this manner, the collection of *criteria* to be considered includes:

- Cr.1. BP contribution to strategic goals
- Cr.2. BP contribution to problems resolution
- Cr.3. BP costs
- Cr.4. BP sizes
- Cr.5. BP life cycle steps
- Cr.6. BP customers

1.3.4.3. Define criteria weighting

To define criteria weights, we used the component *Define criteria weighting by identifying the criterion which must be increased first*. The decision maker chose the most important criterion and affected a value of 100 to it. Then, the decision-maker chose the most important criterion and affected a lower value to it. The same principle was applied recursively until a value was affected to all criteria. Normalisation produced weights as shown in Table IV.1.4.

Table IV.1.4. Criteria Weighting.

Criteria	Cr.1	Cr.2	Cr.3	Cr.4	Cr.5	Cr.6
Value	80	100	20	20	60	50
Weight	0.24	0.30	0.06	0.06	0.19	0.15

1.3.4.4. Evaluate alternatives by measuring

Each BP is measured according to the following criteria: cost, size, life cycle step, and customer type. The results are presented in Table IV.1.5.

Table IV.1.5. BP Measurement Results.

Criteria	BP1	BP2	BP3	BP4	BP5	BP6	BP7	BP8
BP costs (in persons)	4	8	5	29	2	1	3	2
BP sizes (sub-processes number)	6	1	1	5	2	2	3	4
BP life cycle steps (nominal)	st. fun.	st. fun.	st. fun.	reg.	reg.	st. fun.	creat.	reg.
BP customers (nominal)	ext.	int.	ext.	ext.	ext.	int.	int.	int.

1.3.4.5. Evaluate alternatives by estimation

This strategy is applied in order to evaluate the BP according to the BP contribution to strategic goals and to problem resolution. In order to define the partial evaluations, we attribute "1" to BPs, which affected either strategic goals or problems to be solved (cf. Table IV.1.6. for BP contribution to problems revolution; and Table IV.1.7. for BP contribution to strategic goals). The final evaluations are the weighted sums of the partial ones.

Table IV.1.7. BP Contribution to Problems Resolution.

Criteria	Frequency	Threat degree	BP1	BP2	BP3	BP4	BP5	BP6	BP7	BP8
BP contribution to problems resolution			*3*	*4*	*13*	*13*	*4*	*4*	*4*	*26*
Delivery failures	3	3				1				1
Delays of payments	3	3			1					
Lacks of circulating assets	2	2			1		1			1
Supply problems	3	3								1
Exceeding of throughput time	1	2				1				1
Increase of stocks	2	1	1		1					1
Delivery failures	2	2		1		1				1
Delays of payments	3	1						1		
Lacks of circulating assets	3	1							1	

Table IV.1.6. BP Contribution to Strategic Goals.

Criteria	Weights	BP1	BP2	BP3	BP4	BP5	BP6	BP7	BP8
BP contribution to strategic goals		*18*	*5*	*12*	*18*	*2*	*2*	*1*	*9*
Increase of turnover	2	1		1	1				1
Increase of turnover for new customers	2	1		1	1				
Increase of turnover for new products	2	1	1	1	1				
Increase of "in time" delivery	2	1			1				
organization image improving	2	1			1				
Number of partnership increasing	2	1			1				
Increasing of returning customers' share	2			1	1				1
Market part increasing for capacitors	2	1	1		1				1
Market part increasing for filters	2	1		1					
Market part increasing for varistors	2	1		1		1			
Productiveness increasing	1				1				
"Time-to-market" decreasing	1		1						1
Increase of productive capacity use	1				1				1
"In time" supplies number stabilization	1						1		
Number of learned persons	1								1
Conditions of work satisfaction improvement	1						1		
Automation	1							1	

To analyze the influence of BP on strategic goals, the goals are divided into two categories: "results" that concern financial performance and customers, and "leverages" that concern internal processes, learning and growth. Weights were distributed within these categories: 2 – "results" and 1 – "leverages".

The evaluation of BP is summed up in Table IV.1.8.

TableIV.1.8. BP Evaluation Summary.

Criteria	Weights	BP1	BP2	BP3	BP4	BP5	BP6	BP7	BP8
BP contribution to strategic goals (in points)	0,24	18	5	12	18	2	2	1	9
BP contribution to problems resolution (in points)	0,30	3	4	13	13	4	4	4	26
BP costs (in persons)	0,06	4	8	5	29	2	1	3	2
BP sizes (sub-processes number)	0,06	6	1	1	5	2	2	3	4
BP life cycle steps (nominal)	0,19	st. fun.	st. fun.	st. fun.	reg.	reg.	st. fun.	cre-at.	reg.
BP customers (nominal)	0,15	ext.	int.	ext.	ext.	ext.	int.	int.	int.

These data are used for producing the final result by outranking relation of synthesis.

1.3.4.6. Make Decision by Outranking Relation of Synthesis

The given component applied to the case of BP prioritization is shown at Figure IV.1.6.

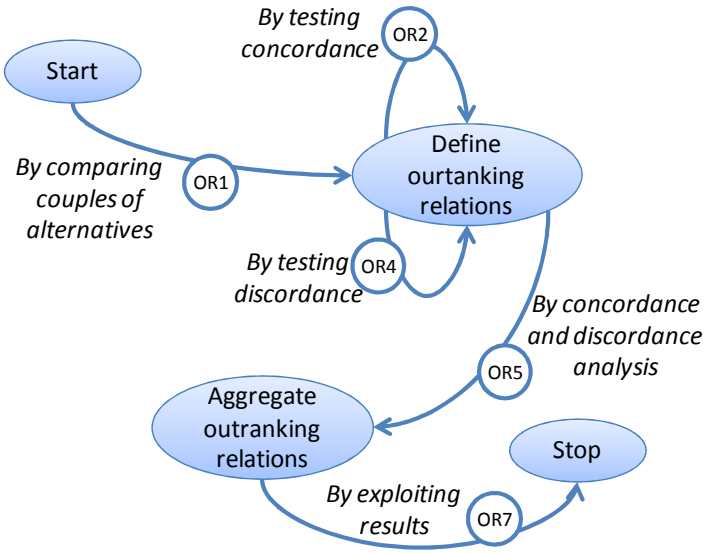


Figure IV.1.6. DM Method Component 'Make Decision by Outranking Relation of Synthesis' in the Case of BPM.

The concordance and discordance matrices developed in our case study are shown in Table IV.1.9. and Table IV.1.10. respectively. Using a threshold of 0,55 to highlight BP in the concordance and discordance tables reveals that BP8 (Manage Financial and Physical Resources) dominates the others

on average without any particular shortcoming in terms of discordance. The three most important business processes are BP1, BP3, and BP8.

Table IV.1.9. Concordance Matrix.

	BP1	BP2	BP3	BP4	BP5	BP6	BP7	BP8
BP1		0,70	0,70	0,51	0,45	0,64	0,64	0,45
BP2	0,49		0,25	0,06	0,54	0,88	0,88	0,15
BP3	0,64	1,00		0,51	0,69	0,88	0,88	0,39
BP4	0,88	0,94	0,94		0,94	0,94	0,94	0,64
BP5	0,70	0,76	0,46	0,40		1,00	0,94	0,40
BP6	0,55	0,76	0,31	0,06	0,66		0,94	0,21
BP7	0,36	0,57	0,12	0,06	0,36	0,51		0,15
BP8	0,55	1,00	0,61	0,55	0,79	0,94	1,00	

Table IV.1.10. Discordance Matrix.

	BP1	BP2	BP3	BP4	BP5	BP6	BP7	BP8
BP1		0,04	0,43	0,43	0,14	0,14	0,04	1,00
BP2	0,76		0,39	0,76	0,25	0,25	0,14	0,96
BP3	0,35	0,00		0,35	0,18	0,18	0,07	0,57
BP4	0,86	0,75	0,82		1,00	1,00	0,89	0,93
BP5	0,94	0,18	0,59	0,94		0,00	0,20	0,96
BP6	0,94	0,18	0,59	0,94	0,50		0,20	0,96
BP7	1,00	0,24	0,65	1,00	0,11	0,11		0,96
BP8	0,53	0,00	0,18	0,53	0,07	0,07	0,00	

1.3.4.7. Prescribe Decision by Validation

A qualitative analysis of this choice revealed that the enterprise agreed with it. A reengineering of the financial and physical resource management processes was thus undertaken.

1.3.5. Conclusions on the Case Study

We have illustrated the application of the MADISE approach for making decisions in the field of the BP prioritization. This example deals with a decision which is strategic and has long-term consequences. In this case, it is important to have a structured DM approach considering multiple factors and shared by different stakeholders.

Thus, the application of the MADISE approach is justified. In addition, it provides more detailed guidelines than the existing approaches. Defining BP priorities with a structured MC method has advantages: (i) time taken to make and realize decisions decreases, thanks to less analysis mistakes in the BP, (ii) expenses decrease, (iii) degree of goals achievement grows by targeting the most

important BPs, and (iv) stakeholders confidence in results and in the overall project grows, owing to their participation in the definition of priorities.

Besides, BP prioritization could be achieved in others contexts, such as: ERP implementation; business continuity plan elaboration; or improvement of Information System strategic alignment. We believe that other case studies in these domains and comparative analyses should be undertaken to fully validate our approach.

1.4. Requirement Engineering

Multiple DM methods exist for resolving DM problems in the field of requirement engineering. We compare our approach with five known requirements prioritization approaches using a comparison framework. In this Section, we present the comparison framework and its application to six approaches (MADISE and five requirements prioritization approaches).

For comparison, we formulate the following hypothesis:

> ***Hypothesis:*** *The DM method family provides a more compete and more flexible way of prioritizing requirements compared to the sample of the well-known requirements prioritization methods.*

1.4.1. Comparison Framework

We compare our approach with existing requirements prioritization techniques according to two aspects: completeness and flexibility.

We consider a technique as complete if it is generic and covers all basic DM activities. We have detailed the main DM steps in order to distinguish the basic DM activities as follows:

- Requirements list identification;
- Requirements list refinement;
- Criteria list definition;
- Criteria list refinement;
- Criteria weighting;
- Requirements evaluation;
- Requirements evaluation refinement;
- Decision-Making;
- Validation;
- Execution.

Requirements list identification. This activity explains how to identify the initial set of requirements to be prioritized (for instance, in [Grosz *et al.*, 1997]).

Requirements list refinement. The initial requirements set may be refined, as requirements can be complementary or exclusive to each other. They can be removed from the list as non-realistic or non-feasible [Roy, 1996] or added to the initial set by searching complementary alternatives.

Criteria list definition. This activity explores how to identify the criteria list for comparing requirements. Criteria can be deduced from the requirements description, from consequences analysis of goal analysis. For instance, goals as criteria are used in [Maiden *et al.*, 2002] for requirements prioritization.

Criteria list refinement. Once the first set of criteria is selected, it can be refined. Criteria can be eliminated as not relevant or added when criteria can complement each other.

Criteria weighting. The By weighting strategy deals with the weights assignment to the decision criteria. Weights are assigned when the engineer wants to define relative importance of criteria. For instance, weights can be assigned by the pair-wise comparison [Saaty, 1980]. The last one is commonly used in the requirements prioritization within AHP [Ruhe *et al.*, 2003].

Requirements evaluation. Requirements must be evaluated according to the criteria list or compared between them. For instance, a pair-wise comparison is present in the AHP method applied for the requirements prioritization [Karlsson *et al.,* 1997] [Ruhe *et al.*, 2003].

Requirements evaluation refinement. Once the requirements are evaluated, their evaluations can be enhanced. For example, a domination analysis can be carried out. An acceptance threshold is established for each criterion. It allows to qualify several requirements as non-acceptable and to remove them from the list. Such a technique is used in the WinWin method for the requirements prioritization [Ruhe *et al.*, 2003].

Decision-Making. The decision is made when partial values (alternatives values according to different criteria) are transformed into an aggregated one. There are three main aggregation approaches: aggregation into a single criterion, outranking approach and interactive approach. Multi-criteria DM method are used on this step [Roy, 2005].

Validation. A DM actor receives a prescription for the decision. If he agrees with results, he validates them. Some metrics (for instance, a consistency index in [Karlsson *et al.* 1997]) can be used to check if the DM result is valid.

Execution. The execution activity deals with the available support for DM. This activity is present in a given requirement prioritization support if a support tool is available in a given approach.

The **completeness** criterion is calculated as a percentage of the available activities composing a given DM method on the total number of the main DM activities.

Flexibility is important when we investigate whether a DM process is predefined or adaptable. Processes must be flexible in order to better match a project following the situation [Rolland, 1998].

Flexibility refers to two criteria: variation and iterativity, both are calculated based on the steps number.

- *Variation points number* represents a number of steps where a choice can be made.
- *Backward steps number* is a number of steps where engineers can return in the requirements prioritization process.
- *Number of steps* is the total number of the main steps.

The variation criterion shows the percentage of steps where a choice between some actions can be made. The iterativity criterion represents the percentage of steps where a backward action can be undertook.

1.4.2. Framework Application

The described frame is applied to the five selected requirements prioritization techniques and to the MADISE approach. The results are presented in Tables IV.1.11 and IV.1.12. for the completeness criterion and the flexibility criteria respectively.

Table IV.1.11. Comparison according to the Completeness Criterion.

Comparison Criteria		***CVA*** [Karlsson *et al.*, 1997]	***PrMatrix*** [Wiegers, 1999]	***RPT*** [Moisiadis, 2002]	***WinWin*** [Ruhe *et al.*, 2003]	***REDEPEND*** [Maiden *et al.*, 2002]	***MADISE***
Requirements list identification		-	-	-	+	-	+
Requirements list refinement		+	+	-	+	-	+
Criteria list definition		-	-	-	-	-	+
Criteria list refinement		-	-	-	-	-	+
Criteria weighting		-	+	+	+	+	+
Requirements evaluation		+	+	+	+	+	+
Requirements evaluation refinement		-	-	+	+	-	+
Decision-Making		+	+	+	+	+	+
Validation		+	-	+	+	-	+
Execution		+	+	-	+	+	-
Available DM activities	(i)	5	5	5	8	4	9
Number of DM activities	(ii)	10	10	10	10	10	10
Percentage	(i)/(ii)	50%	50%	50%	80%	40%	90%

Table IV.1.12. Comparison according to the Flexibility Criteria.

Comparison Criteria		***CVA*** [Karlsson *et al.,* 1997]	***PrMatrix*** [Wiegers, 1999]	***RPT*** [Moisiadis, 2002]	***WinWin*** [Ruhe *et al.*, 2003]	***REDEPEND*** [Maiden *et al.*, 2002]	***MADISE***
Variation points number	(i)	0	0	0	0	0	5
Backward steps number	(ii)	0	1	0	3	0	3
Number of steps	(iii)	5	8	6	7	3	6
Variation	(i)/(iii)	0%	0%	0%	0%	0%	83%
Iterativity	(ii)/(iii)	0%	13%	0%	43%	0%	50%

We show within an example (Cost-Value Approach) [Karlsson *et al.,* 1997], how these tables were fulfilled.

The cost-value approach contains the following main DM activities:

- *Requirements list refinement.* This approach suggests reviewing candidate requirements for ensuring their completeness and correctness.
- *Requirements evaluation.* Actors (users and engineers) express their preferences by pair-wise comparison for defining the relative value and cost of candidate requirements.
- *Decision-Making.* The aggregated value is obtained by AHP application and is used for ranking requirement. The cost-value approach uses also a cost-value diagram in order to assist DM.
- *Validation.* A consistency index is calculated in order to check the result validity.
- *Execution.* A tool is suggested to support the Cost-Value approach.

The completeness criterion is 50%.

The Cost-Value approach contains five steps:

1. Requirements review;
2. AHP's pair-wise comparison in order to assess the relative value of the candidate requirements;
3. AHP's pair-wise comparison in order to assess the relative cost of the candidate requirements;
4. Cost-Value diagram construction;
5. Requirements prioritization based on the Cost-Value diagram and with the stakeholders participation.

None of these steps contains any variation point or backward step. Therefore, the variation criterion is 0% and the iterativity criterion is 0%.

As we can see, the MADISE approach contains all methodological DM activities and does not include the execution of the DM methods. Between the existing requirements prioritization techniques, the WinWin approach is more complete and covers 80% of the DM activities.

Dealing with the flexibility criteria, we can observe that none of the approaches is completely flexible. All existing requirements prioritization techniques follow a predefined set of steps and, in this manner, do not provide a context-aware DM for prioritizing requirements. However, the MADISE approach suggests different actions for carrying out DM activities. Regarding to the iterativity criterion, there are two existing requirements prioritization techniques (Prioritization Matrix and WinWin) which allow to return to previous steps and to adapt DM process to new circumstances. From this viewpoint, the WinWin approach is also flexible as it allows refining the prioritization results by realize iterations.

Finally, according to the three selected criteria, we show that the MADISE approach is the most flexible between the six studied approaches.

1.5. Conclusion

In this Chapter, we have suggested a supporting tool in order to help the usage of the MADISE approach. After that, we have described the results of the MADISE application.

DM Method Family Support Tool. In order to facilitate the usage of our proposal, we have elaborated a tool supporting the DM method family. The DM method family tool (that we called MADISE Repository) supports the following main function: Storage of the DM method family and DM method components. The MADISE Repository is represented as a Web site developed using PHP5 and MySQL.

MADISE Validation. In order to validate the MADISE approach, we have, firstly, applied it to the case of Business Process Management (BPM), and, secondly, compared it with the main DM methods from the Requirements Engineering field. In the first case, the MADISE approach is validated by creating a specific decision-making process for business processes prioritization. In the second case, it is validated by a comparison with existing approaches of requirements prioritization.

Chapter 2: Conclusion

We stated in this thesis that there is plethora of decisions that have to be made in the ISE field. ISE processes are teleological by their nature. That means they contain steps where decisions must be made. Many methods are available to support this decision-making. These methods have different nature and are more or less efficient in function of the situation in which they are applied. There are many works in the ISE research dealing with DM (aiming at resolving DM problems). The main characteristic of these works is that they resolve a DM problem each time. This finding concerns isolated and selective cases of DM methods application in ISE. However, DM methods application in ISE is still limited.

We have started this thesis by motivating the problems of decision-making in IS engineering. After a presentation of the main decision-making concepts, we have analysed the nature of ISE processes in order to show that all processes have steps that could be considered as DM steps. Thus, DM methods could potentially be applied in all processes of IS engineering.

However, DM methods are often complicated, they require specific human skills, additional information resources, and financial means. In each situation, a question must be asked: *Is the application of a DM method justified?* In order to answer this question, we have developed a typology of decisions in IS engineering. This typology includes eleven classification criteria. By checking these criteria, engineers could position their cases and deduce whether complex and time-consuming decision-making is justified or not.

We have then looked for existing approaches dealing with DM methods selection. We have analyzed them in order to identify lacks of DM in IS engineering. Thus, we have established a list of nine problems to be solved for enhancing DM in this field.

Resolution of the identified problems induced us to elaborate an approach for enhancing decision-making in IS engineering that we called *MADISE* – ***MA***ke ***D***ecisions in ***I***nformation ***S***ystems

Engineering. This approach uses the principles of the Method Engineering science and applies them to the DM methods. The MADISE approach is composed of several parts. The first one is the definition of the DM ontology. The second part is the construction of the DM family with the definition of a set of DM components, its contextualization and configuration. The list of publications realized during this thesis is given in Appendix C.

In the following, we show how the proposed approach answers the Research Questions stated in the Introduction chapter; how it considers the presented issues and solves the problems of DM in ISE identified in the State-of-the-Art chapter. We conclude by presenting several directions for future researches based on this thesis works.

2.1. Responses to the Research Questions

We have formulated the main research question as *How methodological knowledge about decision making can be represented to facilitate the inclusion of decision making processes into ISE methods?*

The main research question was separated into four sub-questions for which we made four proposals.

RQ1: How to identify a decision-making situation and to formalize requirements for decision-making?

We have developed an ontology which models and conceptualizes DM knowledge. This ontology allows to enhance and facilitate decision-making as it formalizes DM methods and method components. The ontology also helps to formalize DM requirements in a specific situation in order to help the selection of DM components for constructing situational DM methods.

This ontology may be viewed as an ontological commitment. It represents an agreement to use the specific DM vocabulary, with respect to all identified DM field concepts. The DM ontology helps to share knowledge with and among DM actors.

RQ2: How decision-making methods should be described?

We have studied the main DM methods in order to identify their specific elementary activities. We have then decomposed the methods into components by grouping these activities following the independency and completeness principles. Components have been defined and formalized following a specific component meta-model. Then they have been organized in a method family with the help of the assembly-based approach.

In order to take into account the reuse context of each of these components, we have also defined a typology of characteristics which we have structured in different facets (each considering a special view of a project). These characteristics may be generic (applicable to any project) or specific (strongly coupled to the specific method or the used representation). Each DM component description may then be linked to specific characteristics values which will help their reutilization.

RQ3: How to construct a context-aware decision-making method in a given situation?

We have identified a flexible configuration process to use the context characteristics typology for defining an application method adapted to a given situation. We modeled this process with the MAP formalism in order to keep a high level of flexibility in the process utilization. It can be applied in different IS engineering situations such as the selection of a component for enhancing an existing IS engineering method or a selection of several components for constructing a new one.

Several ways of configuration of the DM family has been defined: either the engineer wants to select a sub-set of the family, specifically adapted to some characteristics, before starting the navigation, or he wants to be guided through the navigation following the context of the situation, in a run-time configuration.

RQ4: Which kind of support can be offered to IS engineers for improving their methodologies by decision-making?

The usage of the DM method family allows the engineer to improve their methods. We have created a tool which represents the repository of the DM method components. Each DM method component describes a particular DM activity and the handled concepts.

In this manner, the MADISE approach allows to guide engineers and stakeholders through decision-making activities.

2.2. Consideration of the Stated Issues

In the Introduction chapter, we have stated that the MADISE approach must consider the five following issues: variability, context-awareness, conflict resolution, goal orientation, and reusability.

Variability Issue: Regarding to the variability issue, the MADISE approach impact is twofold. Firstly, it offers a way to deal with variability by proposing a solution to select a variant in each variation point. Secondly, it uses the different types of variability in order to provide means for elaborating the list of alternatives in a given DM situation.

Context-awareness Issue: The MADISE approach is based on the situational method engineering principles. Thus, it takes into account the context when configuring a DM method in a given DM situation. The description of the DM situation is made using the DM ontology providing different elements to specify it. The DM method components are also described with different context characteristics. Both DM situation and DM components context ensure the context-awareness of the MADISE approach.

Conflict Resolution Issue: The purpose of this thesis is to make transparent decisions by structuring DM situations and processes. This contributes to the resolution of possible conflicts between actors. Moreover, DM actors are involved in DM activities by different ways specified in this thesis. Thus, the

main contribution of the MADISE approach is about helping actors (both individual and collective) in making decisions.

Goal Orientation Issue: There are two ways to consider goals in the MADISE approach. Firstly, by defining DM method components and organizing them within a method family based on the teleological unicity. Secondly, goals are included into the DM concepts as they could be criteria in several DM situations.

Reusability Issue: The DM method family is produced based on the reusability principle as different DM method components are identified from various DM methods. These components represent reusable building blocks used for constructing new DM application methods adapted to concrete situations.

2.3. Resolution of DM Problems in IS Engineering

With regards to the identified DM problems in IS Engineering, the contributions of the MADISE approach are summed up in Table IV.2.1.

Table IV.2.1. Resolution of the DM Problems in the MADISE Approach.

N	DM Problems in ISE	Problems solved in the MADISE Approach	
1	Take into account the problem situation	Yes	Decision-making situations are considered for both DM situation and DM method components context. A contextualization approach is offered in this thesis to allows specifying the context of DM and other methods.
2	Allow a typology of problem characteristics; take into account data diversity	Yes	The characteristics related to the DM situation are organized using the ontology representation (the DM ontology); The characteristics related to the DM method components context are organized within a typology (the typology of the DM context characteristics).
3	Consider all main groups of the existing DM methods and be able to deal with a new one	Yes	The DM method family includes components deduced from the five groups of discrete DM methods. This problem is solved within the ISE field. However, other methods may enrich the given family by applying the suggested method family definition process.
4	Allow selecting of DM method, its better understanding and construction	Yes	This problem is solved by introducing an approach for DM method family configuration. It allows selecting the appropriate DM components and construct DM application methods.
5	Take into account interaction between goals	Partial	There is only one way to address the goal interaction in this thesis: the DM method components derived from the AHP method offer a way to consider goals organized within a hierarchy. Other possible goals' interactions (exclusion, complementarity, etc.) are not taken into account in this approach.

6	Be structured	Yes	Both product (DM ontology and DM method family) and process (contextualization and configuration approaches) parts of the MADISE approach are well structured.
7	Be universal with respect to different application domains	Partial	The MADISE approach is universal in the IS engineering field with discrete decision-making. In order to be applicable to other fields, the DM method family could be enriched by components derived from the continuous DM methods.
8	Permit a capitalization of selection results	No	The capitalization of the DM results is not addressed in this thesis. However, the DM ontology could be adapted to this usage.
9	Suggest a tool facilitating DM methods selection/construction	Partial	The MADISE repository allows storing the DM method components. It could be improved in order to support the configuration of the DM method family. This tool supports the methodological DM guidelines. Another tool implementing the execution of the DM method components may be developed.

2.4. Future Works

As we can see, several above-mentioned problems are not completely addressed in the MADISE approach. We take into account these problems and other ones in order to suggest future works that could be undertaken based on the results of this thesis.

DM Method Family Extension. The existing DM method family could be extended in order to take into account other DM methods. For instance, fuzzy DM methods could be integrated into the DM method family, or the DM method family could be adapted for considering continuous decision-making. The extension of this family could contribute to its expansion to other application domains.

DM Ontology Usage. The DM ontology is used in this thesis for formalizing DM method components, for specifying DM situation and requirements. However, its usage could be extended to other applications. In the future, different usages of DMO must be investigated both in the ISE and non-ISE domains. For instance, the DM ontology could be adapted for the Design Rationale discipline which aims at documenting decisions and their argumentation.

DM Results Capitalization. The MADISE approach considers only one cycle of decision-making. It could be enhanced in order to capitalize: (i) the DM situation and requirements and (ii) the results of the DM method configuration and application. It would allow reusing not only DM method components but also the results of their application. The DM results capitalization would be useful in the case of repetitive and multi-cycles decisions.

DM Method Family Implementation. In fact, the proposed in this thesis approach provides engineers with the methodological guidelines for making decisions. The next step would be to implement the DM method family. It could be done used the Service-oriented Architecture (SOA) and include the following elements: implementation of the DM method components as services, implementation of the configuration mechanism, composition of method components into an application method, and elaboration of a platform for sharing the obtained application methods with the target tools.

Empirical Validation. We suggest also to validate the MADISE approach by performing empirical studies. For instance, a series of interviews with researchers and practitioners of IS engineering could be undertaken in order to evaluate how the usage of the MADISE approach would facilitate decision-making in practice.

Method Family Concept Improvement and Application to other Domains. The concept of the method family could be developed by integrating the notion of variability. The existing concept is almost applied to the DM methods and to the methods of the scenario elicitation in requirements engineering. In the future, the improved concept could be applied for constructing the other methods, such as agile methods, enterprise architecture methods, and so on.

References

[Abrahamsson *et al.*, 2002] Abrahamsson, P., Salo, O., Ronkainen, J., and Warsta, J.: Agile software development methods: Review and analysis. VTT Publications, Espoo (2002)

[Afonotchkin *et al.*, 2009] Afonotchkin A. I. and Mikhalenko D. G., Management Decisions in Economic Systems, (In Russian) Saint-Petersburg, Russia (2009)

[Agerfalk, 2003] Agerfalk, P.J.: Information systems actability: Understanding Information Technology as a Tool for Business Action and Communication. Doctoral dissertation. Dept. of Computer and Information Science, Linköping University, (2003)

[Agerfalk *et al.*, 2007] Agerfalk, P., Brinkkemper, S., Gonzales-Perez, C., Henderson-Sellers, B., Karlsson, F., Kelly, S., Ralyté, J.: Modularization Constructs in Method Engineering: Towards Common Ground?, Panel of ME 07, Springer, Geneva, Switzerland, (2007)

[Aharoni *et al.*, 2007] Aharoni, A., Reinhartz-Berger, I. : Representation of method Fragments, a comparative study, in proceedings ME 07, Springer, Geneva, Switzerland, (2007)

[Akkermans *et al.*, 2006] H. Akkermans and J. Gordijn, Ontology Engineering, Scientific Method and the Research Agenda, in S. Staab and V. Svatek (Eds.), International Conference on Knowledge Engineering and Knowledge Management No15 EKAW 2006 (4248), Lecture Notes in Computer Science Springer, Tcheque Republique, 112-125 (2006)

[Akkermans *et al.*, 2006a] Akkermans, J., Gordijn, J.: What is this science called requirements engineering? In Glinz, M., Lutz, R., eds.: Proceedings 14th IEEE International Conference on Requirements Engineering (RE06), Los Alamitos, CA, IEEE Computer Society (2006)

[Alexander, 2003] Alexander I., Misuse cases help to elicit non-functional requirements, Computing & Control Engineering Journal, 14 (1), (2003), pp. 40-45

[Amyot *et al.*, 2002] Amyot D., Mussbacher G., URN: Towards a New Standard for the Visual Description of Requirements, In proceeding of 3rd Int. WS on Telecommunications and beyond: the broader applicability of SDL and MSC (2002)

[Anton, 1997] Anton A.I., Goal Identification and Refinement in the Specification of Software-Based Information Systems", Ph.D. Dissertation, Georgia Institute of Technology, (Atlanta, USA, 1997)

[Armenise *et al.*, 1993] Armenise, P., Bandinelli, S., Ghezzi, C., Morzenti, A.: A survey and assessment of software process representation formalisms Int. Journal of Software Engineering and Knowledge Engineering, Vol. 3, No. 3 (1993)

[Aydin, 2006] Aydin, M.N. Decision-making support for method adaptation, Ed. Enschede, Netherlands (2006)

[Bajec, 2005] Bajec M., An approach for creating project-specific software development methodologies, TPSE Cairo (2005).

[Ballestero *et al.*, 1998] Ballestero E., Romero C., Multiple criteria decision making and its applications to economic problems, Kluwer Academic Publishers, Netherlands, 1998, 160p.

[Batanov *et al.*, 2007] Batanov D. N., Vongdoiwang W., Using Ontologies to Create Object Model for Object-Oriented Software Engineering, In *Ontologies: A Handbook of Principles, Concepts and Applications in Information Systems*, Sharman R., Kishore R. and Ramesh R., Springer US, 461-487 (2007)

[Baudry *et al.*, 2002] M. Baudry and N. Vincent, Multicriteria decision making, First annual meeting on health science and technology, Tours, France (2002)

[Beck, 2005] Beck, K.: Extreme programming eXplained: embrace change, 2nd ed. The XP Series. Addison Wesley, Boston, MA, USA (2005)

[Berander, 2005] Berander. P.: Requirements Prioritization. In Engineering and Managing Software Requirements, Eds A. Aurum, C. Wohlin, Springer (2005)

[Berge, 1985] Berge C., Graphes, Gauthier-Villars (1985)

[Bessai *et al.*, 2008] Bessai K., Claudepierre B., Saidani O., Nurcan S., Context-aware business process evaluation and redesign. In proceedings of the international workshop BPMDS'08 (2008)

[Bianco *et al.*, 1999] Bianco G., De Antonellis V., Castano S., and Melchiori M., *A Markov Random Field Approach for Querying and Reconciling Heterogeneous Databases*. Proceedings of the 10th DEXA'99, Florence, Italy (1999)

[Boehm, 1988] Boehm, B.: A Spiral model of software development and enhancement. IEEE Computer 21, 5, 61–72 (1988)

[Boonstra, 2008] Boonstra, A., An empirical taxonomy of IS decision-making processes, http://ideas.repec.org/p/dgr/rugsom/04a03.html (2008)

[Bouquet *et al.*, 2003] Bouquet P., Ghidini Ch., Giunchiglia F., Blanzieri E. (2003). Theories and uses of context in knowledge representation and reasoning. Journal of Pragmatics, 35(3) (2003)

[Bouyssous, 2001] D. Bouyssou, Outranking methods, In Encyclopedia of Optimization, (Kluwer, 2001)

[Bouyssous *et al.*, 2000] Evaluation and decision models: a critical perspective, Bouyssous D., Marchant Th., Pirlot M., Perny P., Tsoukias A., Vincke Ph., Kluwer Academic Publishers, USA, 2000, 274 p.

[Bouyssou *et al.*, 2009] Bouyssou D., Marchant T., Perny P., Social choice and multicriteria decision-making, In Decision-making process: Concepts et methods, Wiley, 2009.

[Bradley *et al.*, 2005] Bradley N. A., Dunlop M. D., Toward a multidisciplinary model of context to support context-aware computing. *Human-Computer interaction*, Lawrence Erlbaum Associates (2005)

[Bresciani *et al.*, 2004] Bresciani P., Giorgini P., Giunchiglia F., Mylopoulos J., and Perini A., TROPOS: An Agent Oriented Software Development Methodology, Journal of Autonomous Agents and MultiAgent Systems, 8(3), (2004), pp. 203-236

[Brinkkemper, 1990] Brinkkemper S., Formalisation of Information Systems Modelling, PhD Thesis, Thesis Publishers, Amsterdam, University of Nijmegen (1990)

[Brinkkemper, 1996] Brinkkemper, S.: Method Engineering: engineering of information systems development method and tools, Information and Software Technology, 38(7), (1996)

[Brinkkemper *et al.*, 2001] Brinkkemper, S., Saeki, M., Harmsen, A.F.: A method engineering Language for the description of systems development methods, in proceedings of the conference CAISE 01. Springer Verlag. Interlaken, Switzerland, (2001)

[Bucher *et al.*, 2008] Bucher T., Bajec M., Furlan Š., Kornyshova E., Saidani O., Vavpotiè D., and Žvanut B., On the Application of the ISD Method Engineering Approach in Non-ISD Domains, Institute of Information Management (University of St. Gallen), Working paper, Novembre (2008)

[Castano *et al.*, 1993] Castano S. and De Antonellis V., A Constructive Approach to Reuse of Conceptual Components, Proceedings of Advances in Software Reuse: Selected Papers from the Second International Workshop on Software Reusability, Lucca, Italy (1993)

[Cayla, 2008] Cayla D.: Organizational Learning: A Process Between Equilibrium and Evolution, Journal of Economic Issues 42, 2, 553-560 (2008)

[Chakhar, 2006] Chakhar S. : Cartographie décisionnelle multicritère: formalisation et implémentation informatique, PhD Thesis (In French), Université Paris Dauphine, France (2006)

[Chung *et al.*, 1999] Chung L., Nixon B. A., Yu E., and Mylopoulos J., Non-functional requirements in software engineering, (Kluwer Academic Publishers, 1999)

[Corcho *et al.*, 2005] Corcho O., Fernández M., Gómez-Pérez A., López-Cima A., Building Legal Ontologies with METHONTOLOGY and WebODE, In: Law and the Semantic WebHeidelberg, DE: Springer (2005), pp. 142-157.

[Cossentino *et al.*, 2005] Cossentino M., Seidita V., Composition of a new process to meet agile needs using method engineering. Software Engineering for Large Multi-Agent Systems Vol. III. LNCS Series, Vol. 3390. Springer-Verlag (2005)

[Cossentino *et al.*, 2006] Cossentino M., Gaglio S., Henderson-Sellers B., Seidita V., A metamodelling-based approach for method fragment comparison, Proceedings of the 11th International Workshop on Exploring Modeling Methods in Systems Analysis and Design (EMMSAD), Luxembourg (2006)

[Cossentino *et al.*, 2007] Cossentino M., Gaglio S., Garro A., Seidita V., Method fragments for agent design methodologies: from standardisation to research, *Int. J. of Agent-Oriented Software Engineering*, Vol. 1 (2007), pp. 91-121

[Crowe *et al.*, 1997] Crowe, T.J., Rathi, K., Rolfes, J.D.: Applying a Taxonomy of Business Processes to Identify Reengineering Opportunities, http://www.prosci.com/rathi.htm (1997)

[Dardenne *et al.*, 1993] Dardenne A., Lamsweerde A., and Fickas S., Goal-directed Requirements Acquisition, Science of Computer Programming, 20, (Elsevier, 1993), pp. 3-50

[De Antonellis *et al.*, 1991] De Antonellis V., Pernici B., and Samarati P., *F-ORM METHOD : A methodology for reusing specifications,* in Object Oriented Approach in Information Systems, Van Assche F., Moulin B., Rolland C. (eds), North Holland (1991)

[Deelstra *et al.*, 2004] Deelstra S., Sinnema M., Nijhuis J., Bosch J., COSVAM: A Technique for Assessing Software Variability in Software Product Families, In Proceedings of the 20th IEEE International Conference on Software Maintenance (ICSM'04), Chicago, Illinois, pp. 458-462 (2004)

[Deneckère, 2001] Deneckère R., Approche d'extension de méthodes fondée sur l'utilisation de composants génériques, PhD thesis (In French), University of Paris 1-Sorbonne (2001)

[Deneckère *et al.*, 2001] Deneckère R., Souveyet C., Organising and selecting patterns in pattern languages with Process Maps", *Object Oriented Information Systems (OOIS)*, Calgary, Canada (2001)

[Deneckère *et al.*, 2008] Deneckère R., Ralyté J., and Kraiem N., Modélisation modulaire de méthode: le méta-modèle COMET. In « Ingénierie des méthodes au service des nouvelles tendances de développement des applications informatiques », (In French) N. Kraïem & Y. Jamoussi (Eds.), Centre de Publication Universitaire, Tunis, pp :117-149 (2008)

[Derguech *et al.*, 2010] Derguech W., Vulcu G., Bhiri S., An Indexing Structure for Maintaining Configurable Process Models, in Proceedings of the BPMDS 2010 and EMMSAD 2010, LNBIP 50, Hammamet, Tunisia, pp. 157-168 (2010)

[Decision Deck, 2009] Decision-Deck project, (accessed by January, 2009) http://www.decision-deck.org/ (2009)

[Dey *et al.*, 2001] Dey A., Abowd G., Salber, D., A conceptual framework and toolkit for supporting the rapid prototyping of context-aware applications, *Human-computer Interaction* (Special issue on context-aware computing), 16(2-4), 97-166 (2001)

[Dowson, 1988] Dowson, M.: Iteration in the Software Process, Proc 9th Int. Conf. on Software Engineering (1988)

[Dowson, 1993] Dowson, M.: Software Process Themes and Issues, IEEE Conf. on the Software Process (1993)

[Etien, 2006] Etien A., L'ingénierie de l'alignement: Concepts, Modèles et Processus. La méthode ACEM pour la correction et l'évolution d'un système d'information aux processus d'entreprise, PhD thesis (In French), University of Paris 1-Sorbonne (2006)

[Euler, 1736] Euler L., Solutio problematis ad geometriam situs pertinentis, Commetarii Academiae Scientiarum Imperialis Petropolitanae (1736), 128-140.

[Felix, 1995] Felix R., Fuzzy decision making based on relationships between goals compared with the analytic hierarchy process, In proceedings of the Sixth International Fuzzy Systems Association World Congress, Vol.II, Sao Paulo, Brazil (1995) pp. 253-256

[Fernández-López *et al.*, 1997] Fernández-López M, Gómez-Pérez A, Juristo N, *METHONTOLOGY: From Ontological Art Towards Ontological Engineering*. Spring Symposium on Ontological Engineering of AAAI. Stanford University, California, (1997), pp 33–40

[FIPA, 2003] Method fragment definition, FIPA Document (accessed by November, 2003), http://www.fipa.org/activities/methodology.html (2003)

[Firesmith *et al.*, 2001] Firesmith D., Henderson-Sellers B., The OPEN Process Framework. An Introduction, Addison-Wesley (2001)

[Firesmith, 2006] Firesmith, D.: Method Engineering Using OPFRO, European SEPG, Netherlands (2006)

[Franckson *et al.*, 1991] Franckson M., Peugeot C., Specification of the object and process modeling langage, ESF report n°D122-OPML-1.0 (1994)

[Fuller *et al.*, 1996] R. Fuller and C. Carlsson, Fuzzy multiple criteria decision making: Recent developments, Fuzzy Sets and Systems, 78, (1996), pp. 139-153

[Gomez-Limon *et al.*, 2003] Gómez-Limón J.A., Riesgo L., and Arriaza M., Multi-Criteria Analysis of Factors Use Level: The Case of Water for Irrigation, Proceedings of the 25th International Conference of Agricultural Economists, (2003)

[Gómez-Pérez, 2001] Gómez-Pérez A., Evaluation of ontologies, Workshop on Verification and Validation at DEXA (Database and EXpert System Applications), Vienna, AUTRICHE, vol. 16, n°3, (2001) pp. 391-409

[Gonzales-Perez, 2007] Gonzales-Perez, C.: Supporting Situational Method Engineering with ISO/IEC 24744 and the Work Product Tool Approach. Proceedings of the International IFIP WG8.1 Conference ME 07, Springer, Geneva, Switzerland, (2007)

[Grosz *et al.*, 1997] Grosz, G., Rolland, C., Schwer, S., Souveyet, C., Plihon, V., Si-Said, S., Ben Achour, C., Gnaho, C.: Modelling and Engineering the Requirements Engineering Process: An Overview of the Nature Approach, Requirements Engineering Journal, 2, 115-131 (1997)

[Gruber, 1993] Gruber Thomas R., Toward Principles for the Design of Ontologies Used for Knowledge Sharing, In International Journal Human-Computer Studies 43, p.907-928. Substantial revision of paper presented at the International Workshop on Formal Ontology, March, Padova, Italy (1993)

[Grüninger *et al.*, 1995] Grüninger M., Fox M.S., Methodology for the design and evaluation of ontologies, In: Skuce, D., ed. IJCAI 95 Workshop on basic ontological issues in knowledge sharing. (1995)

[Gu *et al.*, 2004] Gu T., Wang X.H., Pung H.K., Zhang D.Q., An Ontology-based Context Model in Intelligent Environments. In proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference, 270-275 (2004)

[Guidebook, 2001] GUIDEBOOK TO DECISION-MAKING METHODS, Developed for the Department of Energy, by Dennis Baker, Donald Bridges, Regina Hunter, Gregory Johnson, Joseph Krupa, James Murphy, Ken Sorenson, (2001), http://emi-web.inel.gov/Nissmg/Guidebook_2002.pdf

[Guzélian *et al.*, 2007] Guzélian, G., Cauvet, C.: SO2M : Towards a Service-Oriented Approach for Method Engineering, in: the 2007 World Congress in Computer Science, Computer Engineering and Applied Computing, in the proceedings of the international conference IKE'07, Las Vegas, Nevada, USA, (2007)

[Hammer *et al.*, 1993] Hammer M. and Champy J., *Le reengineering: Réinventer l'entreprise pour une amélioration spectaculaire de ses performances*, Dunod, Paris (1993)

[Hanne, 1999] Hanne T., Meta Decision Problems in Multiple Criteria Decision Making. Chapter 6 in T. Gal, T.J. Stewart, T. Hanne (ed) Multiple Criteria Decison Making-Advances in MCDM Models, Algorithms, Theory and Applications. International Series in Operations Research and Management Science, Volume 21. Springer (Kluwer), 1999.

[Harmsen *et al.*, 1994] Harmsen, A.F., Brinkkemper, J.N., Oei, J.L.H.: Situational Method Engineering for information Systems Project Approaches Int. IFIP WG8. 1 Conference in CRIS series: "Methods and associated Tools for the Information Systems Life Cycle" (A-55), North Holland (Pub.), 169-194 (1994)

[Harmsen, 1997] Harmsen F., *Situational method engineering*. Moret Ernst & Young (1997)

[Henderson-Sellers, 2002] Henderson-Sellers, B.: Process meta-modelling and process construction: examples using the OPF. Ann. Software Engineering, 14(1-4), 341-362 (2002)

[Henderson-Sellers *et al.*, 2004] Henderson-Sellers B., Serour M., McBride T., Gonzalez-Perez C., Dagher L., Process construction and customization, Journal of Universal Computer Sciences, 10(3), online journal accessible at http://www.jucs.org (2004)

[Henderson-Sellers *et al.*, 2005] Henderson-Sellers B., Gonzalez-Perez C., McBride T., A meta-model for assessable software development methodologies. Software Quality Journal, 13(2) (2005)

[Henderson-Sellers *et al.*, 2005b] Henderson-Sellers B., Gonzalez-Perez C., Serour M. K., and Firesmith D. G., Method engineering and COTS evaluation. In ACM SIGSOFT Software Engineering Notes (SIGSOFT) 30(4):1-4 (2005)

[Henderson-Sellers, 2006] Henderson-Sellers, B.: SPI – A role for Method Engineering, Proceedings of the 32nd EUROMICRO, SEAA'06, (2006)

[Henderson-Sellers *et al.*, 2007] Henderson-Sellers, B., Gonzalez-Perez, C., Ralyté, J.: Situational Method Engineering: Fragments or Chunks?, proceedings of CAiSE'07 Forum, Trondheim, Norway, (2007)

[Henderson-Sellers *et al.*, 2010] Henderson-Sellers B., Ralyté J., Situational Method Engineering: State-of-the-Art Review. In Journal of Universal Computer Science, vol. 16, no 3 (2010) pp. 424-478

[Hevner *et al.*, 2004] Hevner, A., March, S., Park, J., Ram, S., Design science research in information systems. MIS Quarterly 28(1) (2004) 75–105

[Iacovelli *et al.*, 2008] Iacovelli, A., Souveyet, C., Rolland, C. Method as a Service (MaaS), In Proceedings of the International Conference Research Challenges in Information Science (RCIS) (2008) 371-380

[ISO/IEC 15939, 2002], ISO/IEC 15939: Software Engineering - Software Measurement Process (2002)

[ISO/IEC 24744, 2007] International Standards Organization / International Electrotechnical Commission: Software Engineering. Metamodel for development Methodologies, ISO/IEC 24744, Geneva, (2007)

[Jarke *et al.*, 1992] Jarke M., Pohl K., Information systems quality and quality information systems, In Proc. Of the IFIP 8.2 working conference on the impact of computer-supported techniques on information systems development, Mineapolis, NM (1992)

[Jacobson *et al.*, 1992] Jacobson I., Christerson M., Jonsson P., and Oevergaard G., *Object Oriented Software Engineering: a Use Case Driven Approach*. Addison-Wesley (1992)

[Jarke *et al.*, 1993] Jarke M., Pohl K., Requirements Engineering: An Integrated View of Representation, Process and Domain, Proc. 4th European Software Conf., Springer Verlag (1993)

[Jarke *et al.*, 1994] Jarke, M., Pohl, K., Rolland, C., Schmitt, J. R:. Experienced-Based Method Evaluation and Improvement: A Process Modeling Approach, Int. IFIP WG8. 1 Conf. in CRIS series: Method and associated Tools for the Information Systems Life Cycle, North Holland (1994)

[Jeusfeld *et al.*, 2007] Jeusfeld, M., Backlund, P., Ralyté, J.: Classifying Interoperability Problems for a Method Chunk Repository. I-ESA'07, Funchal, Portugal (2007)

[Kaiya *et al.*, 2006] Kaiya H., Saeki M., Using Domain Ontology as Domain Knowledge for Requirements Elicitation, Requirements Engineering, 14th IEEE International Conference Volume , Issue , 11-15 Sept. 2006 Page(s):189 – 198 (2006)

[Kangas *et al.*, 2001] Kangas A., Kangas J., and Pykäläinen J., Outranking Methods As Tools in Strategic Natural Resources Planning, 35(2), Silva Fennica (2001) pp. 215–227

[Karlsson *et al.*, 1997] Karlsson J., Ryan K.: A Cost–Value Approach for Prioritizing Requirements, IEEE Software (1997)

[Karlsson *et al.*, 2004] Karlsson F., Agerfalk P.J., Method configuration: adapting to situational characteristics while creating reusable assets. Information and Software Technology Vol. 46, Issue 9, 619-633 (2004)

[Karlsson *et al.*, 2004b] Karlsson F., Wistrand K., MC Sandbox - Tool Support for Method Configuration, The Ninth CAiSE/IFIP 8.1/EUNO International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD'04), June 7-8 2004, Riga, Latvia (2004) 199-210

[Karlsson, 2005] Karlsson, F.: Method Configuration: Method and Computerized Tool Support. Doctoral dissertation. Dept of Computer and Information Science. Linköping University. (2005)

[Karlsson *et al.*, 2005] Karlsson F., Agerfalk P.J., Method-User-Centred Method Configuration, In Proceedings of Situational Requirements Engineering Processes - Methods, Techniques and Tools to Support Situation-Specific Requirements Engineering Processes (SREP'05), Paris, France, August 29 – September 30, (2005)

[Kast, 2002] Kast Robert, La théorie de la décision. – Nouv. éd. – Paris : Découverte, 2002.

[Keeney *et al.*, 1993] R.L. Keeney and H. Raiffa, Decisions with Multiple Objectives: Preferences and Value Trade-Offs, (Cambridge University Press, 1993)

[Keeney, 1999] R.L. Keeney, Foundations for Making Smart Decisions, IIE Solutions, 31, No. 5, (1999), pp. 24-30

[Kirsch Pinheiro *et al.*, 2008] Kirsch Pinheiro M., Vanrompay Y., Berbers Y., Context-aware service selection using graph matching. In proceedings of ECOWS 2008, vol. 411 (2008)

[Kou *et al.*, 2009] Kou, Gang, Peng, Yi, A Bibliography Analysis of Multi-Criteria Decision Making in Computer Science (1989-2009), In Cutting-Edge Research Topics on Multiple Criteria Decision Making, Communications in Computer and Information Science, Volume 35. Springer Berlin Heidelberg (2009) p. 68-71

[Kronkof, 1993] Kronlof K., Method integration, concepts and case studies, Wiley series in sofware based systems, John wiley and sons Ltd. (1993)

[Kraiem *et al.*, 2008] Kraiem N. and Jamoussi Y., L'ingénierie des méthodes au service des nouvelles tendances de développement des applications informatiques, Avec la collaboration des professeurs C. Rolland and H. Ben Ghezala, Centre de Publication Universitaire, Tunis (2008)

[Kruchten, 1998] Kruchten, P. The Rational Unified Process. An introduction, Addison-Wesley (1998)

[Laaribi, 2000] Laaribi A., SIG et analyse multicritère, Paris, Hermès Science Publications, 2000, 190p.

[Loucopoulos *et al.*, 1995] Loucopoulos, P., Kavakli, E.: Enterprise Modelling and the Teleological Approach to Requirements Engineering. International Journal of Intelligent and Cooperative Information Systems (1995)

[Lukitcheva *et al.*, 2009] Lukitcheva L. I. and Egorychev D. N., Management Decisions, (In Russian) Moscow, Russia (2009)

[Lyytinen *et al.*, 1989] Lyytinen S. K., Tahvainen V-P., Modeling CASE Environments in systems Work, In Proceedings of the CASE'89 conference, Kista, Sweden (1989)

[Maiden *et al.*, 2002] Maiden, N.A.M., Pavan, P., Gizikis, A., Clause, O., Kim, H., and Zhu X, Integrating Decision-Making Techniques into Requirements Engineering, REFSQ'02, Essen Germany (2002)

[Malcolm, 1967] Malcolm, N.: Explaining Behavior, The Philosophical Review 76, 1, 97-104 (1967)

[Martín *et al.*, 2003] Martín M., Olsina L., Towards an Ontology for Software Metrics and Indicators as the Foundation for a Cataloging Web System, In Proceedings of the First Conference on Latin American Web Congress (LA-WEB 2003), EEE Computer Society, Washington, DC, USA, 103-113 (2003)

[Mazur *et al.*, 2000] Mazur I., Shapiro V., Titov S., and Elkina L., (2000), *Companies restructuring*, (In Russian) Moscow, Ed. High school (2000)

[Mirbel *et al.*, 2002] Mirbel I., De Rivieres V., Adapting Analysis and Design to Software Context : The jecko Approach, In 8th International Conference on Object Orirented Information Systems (2002)

[Mirbel *et al.*, 2006] Mirbel, I. and Ralyte, J., Situational method engineering: combining assembly-based and roadmap-driven approaches, Requirement Engineering Journal, 11(1), (2006)

[Mirbel, 2008] Mirbel I., Contributions à la modélisation, la réutilisation et la flexibilité des systèmes d'information. HDR thesis (in French), Nice University (2008)

[Moisiadis, 2002] Moisiadis, F., The fundamentals of prioritizing requirements, Systems Engineering: Test and Evaluation Conference, Sydney, October (2002)

[Moisiadis, 2005] Moisiadis F., "A Framework for Prioritizing Use Case", Joint Research Centre for Advanced Systems Engineering, Australia, 2005

[Mohabbati *et al.*, 2011] Mohabbati B., Hatala M., Gasevic D., Asadi M., and Boskovic M., Development and configuration of service-oriented systems families, In Proceedings of the 2011 ACM Symposium on Applied Computing (SAC'11), New York, USA (2011)

[Mustajoki *et al.*, 2005] Mustajoki J., Hamalainen R.P., Salo A., "Decision Support by Interval SMART/SWING – Incorporating Imprecision in the SMART and SWING Methods", Decision Sciences, 36(2):317-339, 2005.

[Nehan *et al.*, 2007] Nehan, Y.-R., Deneckère, R.: Component-based Situational Methods: A framework for understanding SME, in IFIP, Volume 244, Situational Method Engineering: Fundamentals and Experiences, Switzerland, (2007)

[NgoThe *et al.*, 2005] Ngo-The, A., Ruhe, G.: Decision Support in Requirements Engineering, In Engineering and Managing Software Requirements, Ed. By A. Aurum and C. Wohlin 267-286 (2005)

[Offen, 2002] OFFEN Ray, Domain understanding is the key to successful system development, Requirements engineering, vol. 7, no3, pp. 172-175, Springer, London, ROYAUME-UNI (2002)

[Olle *et al.*, 91] Olle W. H. J., Macdonald I.G., Rolland C., Sol H.G., Assche F.J.M.V., Verrijn-Stuart A.A., Information Systems Methodologies, Addison-Wesley (1991)

[Olson *et al.*, 2000] Olson D.L., Mechitov A., Moshkovich H., Learning aspects of decision aids, In proceedings of 15th international conference on MCDM'00, Ankara, Turkey (2000) pp. 41-48

[OMG, 2006] Object Management Group (OMG): Meta Object Facility (MOF) v2.0, http://www.omg.org/spec/MOF/2.0/, (2006)

[Ommering, 2002] Ommering, R.V., Building product populations with software components, 24th International Conference on Software Engineering, Orlando, Florida (2002)

[Osterweil *et al.*, 2010] Osterweil L. J., Clarke L.C., Supporting negotiation and dispute resolution with computing and communication technologies, FoSER 2010 (2002) pp. 269-272

[Ozernoy, 1996] Ozernoy V.M., Some fundamental problems in the selection and justification of discrete alternatives MCDM methods, In Proceeding of 11th international conference on MCDM'96, Coimbra, Portugal (1996) pp. 103-112

[Papadacci *et al.*, 2005] Papadacci, E., Salinesi, C., and Sidler, L. Panorama des approches d'arbitrage dans le contexte de l'urbanisation du SI, Etat de l'art et mise en perspective des approches issues du monde de l'ingénierie des exigences. Numéro spécial de la revue ISI «Méthodes Avancées de Développement des SI» Vol 10, N°6, pp 11-30 (2005)

[Papadacci, 2008] Papadacci E., Evaluation qualitative et quantitative de scénarios d'évolution de l'organisation et de son système d'information, PhD thesis (In French), University of Paris 1-Sorbonne (2008)

[Peffers *et al.*, 2006] Peffers K., Tuunanen T., Gengler C. E., Rossi M., Hui W., Virtanen W., Braggev J., The Design Science Research Process: A Model for Producing and Presenting Information Systems Research, In DESRIST 2006 Conference proceedings, Claremont, USA (2006).

[Pohl *et al.*, 2005] Pohl K., Böckle G., van der Linden F., Software Product Line Engineering – Foundations, Principles, and Techniques. Springer, Heidelberg (2005)

[Pomerol *et al.*, 1993] Pomerol J.-C., Barba-Romero S., Choix multicritère dans l'entreprise : Principes et pratique, Hermes, France (1993)

[Poyhonen *et al.*, 2001] Poyhonen M., Hamalainen R.P., "On the convergence of multiattribute weighting methods", European Journal of Operational Research, 129(3):569-585, March 2001.

[Prakash, 1994] Prakash N., A process view of methodologies, 6th international conference on advanced information systems engineerings, In Proceedings of the Conference CAISE'94, Springer Verlag (1994)

[Prakash, 1999] Prakash, N., *On Method Statics and Dynamics. Information Systems*. Vol.34, No.8, pp 613-637 (1999)

[Prakash *et al.*, 2002] Prakash N. and Bhatia MPS,Generic Models for Engineering Methods for Diverse Domains, Advanced Information Systems Engineering. Pidduck A.B., Mylopoulos J., Woo C.C., Ozsu M.T. (eds.) CaiSE 2002, LNCS 2348, pp. 612-625 (2002)

[Prat 1997] 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, pp. 145-156, June 1997.

[Prat 1999] N. Prat, *Réutilisation de la trace par apprentissage dans un environnement pour l'ingénierie des processus*, Thèse de doctorat en informatique de l'université Pris 1, Le 3 février 1999

[Process Classification Framework, 1996] Process Classification Framework http://www.apqc.org/ Accessed by the February 28, (2006)

[Prieto-Diaz *et al.*, 1987] Prieto-Diaz R. and Freeman P., *Classifying Software for reusability*, IEEE Software, Vol. 4, N° 1, January (1987)

[Ralyté 2001] Ralyte J. Method chunks engineering, PhD thesis, University of Paris 1-Sorbonne, (2001)

[Ralyté *et al.*, 2001] Ralyte, J., Rolland, C., An approach for method reengineering. In proceedings of the 20th International Conference on Conceptual Modeling (ER'01), Yokohama, Japan, November 2001. H. Kunii, S. Jajodia, A. Solvberg (Eds.), LNCS 2224, Springer-Verlag, 471-484. (2001)

[Ralyté *et al.*, 2001b] Ralyte, J., Rolland, C., An Assembly Process Model for Method Engineering, *Proceedings of the 13th CAISE'01,* Interlaken , Switzerland 267-283 (2001)

[Ralyté *et al.*, 2003] Ralyté, J., Deneckere, R., Rolland, C.: Towards a Generic Model for Situational Method Engineering, in proceedings of the CAISE'03, Springer Verlag, Velden, Austria (2003)

[Ralyté, 2004] J. Ralyté, Towards Situational Methods for Information Systems Development: Engineering Reusable Method Chunks. Proc. of ISD'04, Vilnius, Lithuania, 271-282 (2004)

[Ralyté *et al.*, 2006] Ralyté J., Backlund P., Kühn H., Jeusfeld M. (2006). Method Chunks for Interoperability. *Proceedings of the 25th International Conference on Conceptual Modelling (ER'2006).* Tucson , Arizona , USA , Springer-Verlag, (2006) pp. 339-353

[Ralph *et al.*, 2008] Ralph, P., Wand, Y.: A Teleological Process Theory of Software Development, in Proceedings of JAIS Theory Development Workshop. Sprouts: Working Papers on Information Systems, 8, 23 (2008)

[Rebstock *et al.*, 2008] Michael Rebstock, Janina Fengel and Heiko Paulheim, Ontologies-Based Business Integration, 268 p., Springer (2008)

[Reinhartz-Berger *et al.*, 2004] Reinhartz-Berger I., Sturn A., Applying the Application-based Domain Modeling Approach to UML Structure Views, ER'2004, Springer, Shanghai, China, 766-779 (2004)

[Reinhartz-Berger *et al.*, 2010] Reinhartz-Berger I., Sturm A., Software variability – Definitions, classifications, and guidelines, Tutorial for ER'2010, Vancouver, Canada (2010)

[Rey *et al.*, 2002] Rey G., Coutaz J., Le Contexteur : une abstraction logicielle pour la réalisation de systèmes interactifs sensibles au contexte. *IHM'02*, (In French) 105-112 (2002)

[Robson *et al.*, 1996] Robson M. and Ullah P., *A Practical Guide to Business Process Reengineering*, Gower Publishing Limited (1996)

[Rolland *et al.*, 1992] Rolland C., Cauvet C., Object-Oriented Conceptual Modelling, CISMOD'92, International Conf. on Management of Data, Bangalore (1992)

[Rolland *et al.*, 1996] Rolland C, Prakash N.; A proposal for context specific method engineering; Proc. IFIP WG8.1 Int. Conf. On "Method Engineering", Chapman&Hall (Pub.), Atlanta, USA (1996)

[Rolland 1997] Rolland C., A Primer for Method Engineering, in Proceedings of the Conference INFORSID (INFormatique des ORganisations et Systèmes d'Information et de Décision), Toulouse, France, June (1997)

[Rolland, 1998] Rolland, C., A Comprehensive View of Process Engineering, Proceedings of the conference CAISE, Pisa, Italy, Springer-Verlag (1998)

[Rolland *et al.*, 1998] Rolland, C., Plihon, V., Ralyté, J.: Specifying the reuse context of scenario method chunks, in the proceedings of the international conference. CAiSE'98, Pise, (1998)

[Rolland *et al.*, 1998b] Rolland, C., Ben Achour, C., Cauvet, C., Ralyte, J., Sutcliffe, A., Maiden, N.M., Jarke, M., Haumer, P., Pohl, K., Dubois, E. and Heymans, P.: A Proposal for a Scenario Classification Framework, Requirements Engineering Journal (1998)

[Rolland *et al.*, 1999] Rolland, C., Prakash, N., Benjamen, A., A Multi-Model View of Process Modelling. Requirements Engineering. Volume 4, Number 4. Springer-Verlag London Ltd (1999)

[Rolland *et al.*, 2000] Rolland, C., Nurcan, S., Grosz G.: A Decision Making Pattern for Guiding the Enterprise Knowledge Development Process, Journal of Information and Software Technology, Elsevier, 42, 313 - 331 (2000)

[Rolland *et al.*, 2001a] Rolland, C. and Prakash N. Matching ERP System Functionality to Customer Requirements. In Procs of the 5th IEEE International Symposium on Requirements Engineering, Toronto, Canada. August 27-31, 2001.

[Rolland *et al.*, 2001b] Rolland C., Ralyté J., An Assembly Process for Method Engineering, Proc. of the 13th CAISE, Springer, 267-283 (2001)

[Rolland *et al.*, 2004] Rolland C., Ralyté J., Ayed M., Construction the Lyee method with a method engineering approach, Knowledge-Based System, 17 (2004)

[Rolland, 2005] Rolland, C., L'ingénierie des méthodes : une visite guidée *(in French:* Method Engineering: A Guided Visit), e-TI - la revue électronique des technologies d'information, 1, http://www.revue-eti.netdocument.php?id=726 (2005)

[Rolland, 2007] Rolland, C., Method Engineering : Achievements, Trends & Challenges, *Situational Method Engineering (ME)*, Invited talk, Genève, Suisse, (2007)

[Rolland *et al.*, 2007a] Rolland C., Prakash N., On the Adequate Modeling of Business Process Families, *Workshop on Business Process Modelling, Development, and Support (BPMDS)*, Held in conjunction with 19th International Conference on Advanced Information Systems Engineering (CAiSE'07)., Trondheim, Norway (2007)

[Rolland *et al.*, 2007b] Rolland C., Prakash N., Kaabi R. S., Variability in Business Process Families, *Information Resources Management Association (IRMA)*, 2007.

[Rolland, 2009] Rolland C., Method engineering: towards methods as services. Software Process: Improvement and Practice 14(3): 143-164 (2009)

[Rolland, 2011] Rolland C., De la modélisation conceptuelle à l'ingénierie des exigences, Techniques de l'ingénieur, February (2011)

[Rosemann *et al.*, 2006] Rosemann M., Recker J., Context-aware process design: exploring the extrinsic drivers for process flexibility. In proceedings of workshops and doctoral consortium in the 18th international conference on advanced information systems engineering. Luxembourg: Namur University Press., 149-158 (2006)

[Roy *et al.*, 1993] Roy B., Bouyssous D., Aide Multicritère à la Décision: Méthodes et Cas, Economica, Paris (1993)

[Roy, 1985] Roy B., *Méthodologie multicritère d'aide à la décision*, Paris, Economica, (1985)

[Roy, 1996] Roy B., Multicriteria Methodology for Decision Aiding, Dordrecht, Kluwer Academic Publishers (1996)

[Roy, 2005] Roy, B. Paradigms and challenges, Book chapter, In Multiple Criteria Decision Analysis - State of the Art Survey, Springer. editor(s) J. Figueira, S. Greco, M. Ehrgott, (2005) 3-24

[Roy *et al.*, 1993] Roy, B. and Bouyssou, D. Aide multicritère à la décision: Méthodes et cas, Ed. Economica, France (1993)

[Royce, 1970] Royce, W. W.: Managing the development of large software systems: concepts and techniques. In Proceedings of Wescon (1970)

[Ruhe, 2003] Ruhe, G.: Software Engineering Decision Support – Methodology and Applications. In: Innovations in Decision Support Systems (Ed. by Tonfoni and Jain), International Series on Advanced Intelligence, Vol. 3, 143-174 (2003)

[Ruhe *et al.*, 2003] Ruhe, G., Eberlein, A., Pfahl, D.: Trade-off Analysis for Requirements Selection, International Journal of Software Engineering and Knowledge Engineering, 13, 4, 345-366 (2003)

[RUP, 2007] Rational Unified Process, Electronic Resource (accessed by June, 2007) http://www-306.ibm.com/software/awdtools/rup/ (2007)

[Saaty, 1980] Saaty T.L., The Analytic Hierarchy Process, NY, McGraw Hill (1980)

[Sachdeva *et al.*, 2005] Sachdeva N. and Joshy J., *On demand business process life cycle, Part 8: Business process monitoring – Create key performance indicators*, http://www-128.ibm.com/developerworks/webservices/library/ws-odbp8/ October 19, (2005)

[Saeki, 2003] Saeki, M. Embedding Metrics into Information Systems Development Methods: An Application of Method Engineering Technique. Proceedings of the CAISE'03, Klagenfurt/Velden, Austria, 374-389 (2003)

[Salber *et al.*, 1998] Salber, D., Dey, K. A., Abowd, D. G., "Ubiquitous Computing : Defining an HCI research agenda for an emerging interaction paradigm". In Georgia Tech GVU technical report, 1998.

[Salinesi *et al.*, 2011] Salinesi C., Mazo R., Djebbi O., Diaz D., Lora-Michiels A. Constraints: the Core of Product Line Engineering. Fifth IEEE International Conference on Research Challenges in Information Science (RCIS), IEEE Press, Guadeloupe-French West Indies, France, May 19-21 (2011)

[Sánchez *et al.*, 2007] Sánchez D. M., Cavero J. M. and Martínez E. M., The Road Toward Ontologies, In *Ontologies: A Handbook of Principles, Concepts and Applications in Information Systems*, Sharman R., Kishore R. and Ramesh R., Springer US, 3-20 (2007)

[Santos *et al.*, 2010] Santos E., Pimentel J., Castro J., Sanchez J., Pastor O., Configuring the Variability of Business Process Models Using Non-Functional Requirements, in Proceedings of the BPMDS 2010 and EMMSAD 2010, LNBIP 50, Hammamet, Tunisia, pp. 274-286 (2010)

[Schafer, 2001] Schafer R., "Rules for Using Multi-Attribute Utility Theory for Estimating a User's Interests", workshop on Adaptivity and User Modelling in Interactive Systems, University of Dortmund, Germany, October (2001)

[Schilit *et al.*, 1994] Schilit B., Adams N. & Want R., Context-aware computing applications., In proceedings of the IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'94), Santa Cruz, CA, US, 89-101 (1994)

[Seidita *et al.*, 2004] Seidita V., Cossentino M. and Gaglio S., A repository of fragments for agent system design, In Proceedings of the 7th WOA 2006 Workshop From Objects to Agents, Vol. 204 (2006)

[Seligmann *et al.*, 1989] Seligmann, P.S., Wijers, G .M., Sol, H.G.: Analysing the structure of IS methodologies, an alternative approach, Proceedings of the 1st Dutch conference on Information Systems, Amersfoort, The Netherlands, (1989)

[Simon, 1960] Simon, H.: The New Science of Management Decision, Harper&Row (1960)

[Sinnema *et al.*, 2004] Sinnema M., Deelstra S., Nijhuis J., Bosch J., COVAMOF: A Framework for Modeling Variability in Software Product Families, Proceedings of the Third Conference Software Product Line Conference (SPLC 2004), Springer Verlag LNCS 3154, pp. 197-213 (2004)

[Si-Said *et al.*, 1997] Si-Said S., Grosz G., and Rolland C., Mentor, A computer Aided Requirements Engineering Environment, International Conference on Advanced information Systems (1997)

[Shemetov *et al.*, 2009] Shemetov P. V., Radionov V. V., Tcherednikova L. E., and Petukhova S. V., Management Decisions: Technology, Methods, Tools, (In Russian) Moscow, Russia (2009)

[Souveyet *et al.*, 2008] Souveyet, C., Iacovelli, A.: Method as a Service (MaaS), Submitted to the conference RCIS'08, (2008)

[Taylor, 1964] Taylor, Ch.: The Explanation of Behaviour. London: Routledge (1964)

[Thompson *et al.*, 2001] Thomphson, J. and Heimdalh, M.P., 2001. Extending the Product Family Approach to support n-Dimensional and Hierarchical Product Lines. RE'01, Toronto, Canada (2001)

[Ulengin *et al.*, 2000] Ulengin, F., Topcu, Y.I., Sahin, S.O.: An artificial neural network approach to multicriteria method selection. In proceedings of 15th international conference on MCDM'00, Ankara, Turkey 101-110 (2000)

[UML, 2011] Introduction to OMG's Unified Modeling Language (accessed by June, 2011), http://www.omg.org/gettingstarted/what_is_uml.htm

[Uschold *et al.*, 1995] Uschold M., King M., Towards a Methodology for building ontologies . In: Skuce D, ed. IJCAI'95m Workshop on Basic Ontological Issue in Knowledge Sharing. Montreal (1995)

[Van Gurp, 2000] Van Gurp, J., Variability in Software Systems, the key to Software Reuse. Licentiate Thesis, University of Groningen, Sweden (2000)

[Van Gurp *et al.*, 2001] Van Gurp J., Bosch J., Svahnberg M., On the Notion of Variability in Software Product Lines, in Proceedings of the Working IEEE/IFIP Conference on Software Architecture (WICSA'01) (2001)

[Van Slooten *et al.*, 1996] Van Slooten K., Hodes B., Characterising IS development projects. In proceedings of the IFIP WG8.1 Conference on Method Engineering (1996)

[VandeVen *et al.*, 1995] Van de Ven A.H., Poole, M.S.: Explaining Development and Change in Organizations, The Academy of Management Review, Vol. 20, No. 3, 510-540 (1995)

[Veblen1898] Veblen, Th.: Why is Economics not an Evolutionary Science? The Quarterly Journal of Economics 12, 4, 373-397 (1898)

[Vincke, 1989] Vincke P, L'aide multicritère à la decision, Edition Ellipses, Edition de l'Université de Bruxelles, Bruxelles, Belgique (1989)

[Vincke, 1995] Vincke Ph., A short note on a methodology for choosing a decision-aid method, Advances in Multicriteria Analysis, Edited by P.M. Pardalos, Y. Siskos and C. Zopounidis, Kluwer Academic Publishers, Netherlands (1995) pp. 3-7

[Wiegers, 1999] K. Wiegers, First Things First: Prioritizing Requirements, Software Development, vol. 7, no. 9, (1999)

[Wieringa *et al.*, 2006] Wieringa, R., Maiden, N., Mead, N., Rolland, C.: Requirements engineering paper classification and evaluation criteria: A proposal and a discussion. Requirements Engineering 11(1) (2006) 102–107

[Wistrand *et al.*, 2004] Wistrand, K., Karlsson, F.: Method components: Rationale revealed, In proceedings of The 16th International Conference on Advanced Information Systems Engineering (CAiSE 2004), Springer-Verlag. Riga, Latvia (2004)

[Zopounidis, 1997] C. Zopounidis, Décisions financières et analyse multicritère, Encyclopédie de Gestion, (Ed. Economica, Paris, 1997), pp. 915-925

Author References

[Deneckère *et al.*, 2011a] Deneckère R., Kornyshova E., and I. Rychkova, Des lignes de processus aux familles de processus, *Informatique des Organisations et Systèmes d'Information et de Décision (INFORSID)*, Lille, France, (2011)

[Deneckère *et al.*, 2011b] Deneckère R., Kornyshova E., Processus téléologique et variabilité : Utilisation de la sensibilité au contexte, *Ingénierie des Systèmes d'Information (ISI)*, Lavoisier, Paris, France, 16:1, (2011), pp. 61 – 88

[Kornyshova *et al.*, 2011a] Kornyshova E., Deneckère R., and Claudepierre B. Towards Method Component Contextualization, *International Journal of Information System Modeling and Design (IJISMD)*, To be published, (2011)

[Kornyshova *et al.*, 2011b] Kornyshova E., Deneckère R., and Rolland C. Method Families Concept: Application to Decision-Making Methods, *Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD)*, London, United Kingdom, (2011)

[Kornyshova *et al.*, 2011c] Kornyshova E., Deneckère R. Famille de méthodes : une approche de construction de méthodes situationnelles, *Informatique des Organisations et Systèmes d'Information et de Décision (INFORSID)*, Papier court, Lille, France, (2011)

[Deneckère *et al.*, 2010a] Deneckère R., Kornyshova E. Process Line Configuration: an Indicator-based Guidance of the Intentional Model MAP, *Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD)*, Hammamet, Tunisie, (2010)

[Deneckère *et al.*, 2010b] Deneckère R., Kornyshova E. La variabilité due à la sensibilité au contexte dans les processus téléologiques, *Informatique des Organisations et Systèmes d'Information et de Décision (INFORSID)*, Marseille, France, (2010)

[Kornyshova *et al.*, 2010a] Kornyshova E., Deneckère R., and Claudepierre B. Contextualization of method components, *International Conference on Research Challenges in Information Science (RCIS)*, Ed. IEEE, ISBN #978-1-4244-4840-1, Nice, France, (2010)

[Kornyshova *et al.*, 2010b] Kornyshova E., Deneckère R. Decision-Making Ontology for Information System Engineering, *International Conference on Conceptual Modeling (ER)*, Vancouver, Canada, November (2010)

[Deneckère *et al.*, 2009a] Deneckère R., Kornyshova E., and Rolland C. Enhancing the Guidance of the Intentional Model MAP: Graph Theory Application, *International Conference on Research Challenges in Information Science (RCIS)*, Fès, Morocco, (2009)

[Kornyshova *et al.*, 2009a] Kornyshova E., Salinesi C., and Deneckère R. Which Method to Support Multicriteria Decision Making Systematically in IS Engineering?, *Systems Research Forum Journal (SRF)*, 3:1, (2009), pp. 15 – 24

[Bucher *et al.*, 2008a] Bucher T., Bajec M., Furlan Š., Kornyshova E., Saidani O., Vavpotiè D., and Žvanut B. On the Application of the ISD Method Engineering Approach in Non-ISD Domains, *Institute of Information Management (University of St. Gallen)*, Working paper, (2008)

[Deneckère *et al.*, 2008b] Deneckère R., Iacovelli A., Kornyshova E., and Souveyet C. From Method Fragments to Method Services, *Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD)*, Montpellier, France, (2008)

[Kornyshova *et al.*, 2008a] Kornyshova E., Salinesi C. Selecting MCDM Techniques: State of the Art, *International Journal of Information Technology and Intelligent Computing (IT&IC)*, (2008)

[Kornyshova *et al.*, 2008b] Kornyshova E., Deneckère R., and Salinesi C. Using Multicriteria Decision-Making to Take into Account the Situation in System Engineering, *International Conference on Advanced information Systems Engineering (CAISE)*, In Forum Proccedings, Montpellier, France, (2008)

[Kornyshova *et al.*, 2008c] Kornyshova E., Deneckère R., and Salinesi C. Improving Software Development Processes with Multicriteria Methods, *Model Driven Information Systems Engineering: Enterprise, User and System Models (MoDISE-EUS)*, Montpellier, France, (2008)

[Kornyshova *et al.*, 2008d] Kornyshova E., Deneckère R., and Salinesi C. Improving Software Development Processes with Multicriteria Methods, *Méthodes Avancées de Développement des Systèmes d'Information (MADSI)*, Fontainebleau, France, (2008)

[Kornyshova *et al.*, 2008e] Kornyshova E., Salinesi C. Introducing Multicriteria Decision Making into Software Engineering, *INSIGHT (INSIGHT)*, INCOSE, 11:3 (2008)

[Kornyshova *et al.*, 2007a] Kornyshova E., Salinesi C. MCDM Techniques Selection Approaches: State of the Art, *IEEE Symposium on Computational Intelligence in Multicriteria Decision Making (MCDM)*, Honolulu, HI, USA, (2007)

[Kornyshova *et al.*, 2007b] Kornyshova E., Salinesi C. Business Process Priorisation with Multicriteria Methods: Case of Business Process Reengineering, *International Conference on Enterprise Information Systems (ICEIS)*, Funchal, Portugal, (2007)

[Kornyshova *et al.*, 2007c] Kornyshova E., Deneckère R., and Salinesi C. Method Chunks Selection by Multicriteria Techniques: an Extension of the Assembly-based Approach, *IFIP, volume 244 (Situational Method Engineering: Fundamentals and Experiences) (ME)*, Geneva, Switzerland, (2007)

[Kornyshova, 2007d] Kornyshova E. Développement d'une démarche visant à introduire l'aide à la décision multicritère dans l'ingénierie des systèmes, *Association Française d'Ingénierie Système (AFIS)*, Poster, Nancy, France, (2007)

[Salinesi *et al.*, 2006a] Salinesi C., Kornyshova E. Choosing a Prioritization Method – Case of IS Security Improvement, *International Conference on Advanced information Systems Engineering (CAISE)*, In Forum Proceedings, Luxembourg, Luxembourg, (2006), pp. 51 - 55