Representation of Method Fragments: A Domain Engineering Approach

Iris Reinhartz-Berger[1] and Anat Aharoni[1]

[1] Department of Management Information Systems,
University of Haifa, Haifa 31905, Israel
iris@mis.haifa.ac.il, anatah@mis.haifa.ac.il

Abstract. Abstract. The discipline of situational method engineering (SME) promotes the idea of retrieving, adapting, and tailoring fragments, rather than complete methodologies, to specific situations. In order to succeed in creating good methodologies that best suit given situations, fragment representation and cataloguing are very important activities. We introduce a visual SME approach, whose roots are in domain engineering. This approach relies on the Application-based DOmain Modeling (ADOM) approach, which provides a framework for representing both applications and domains and validating them each against the other. Furthermore, the proposed ADOM-based approach aims at supporting all the SME-related activities, while in this paper we focus only on its fragment representation and cataloguing parts. The main advantages of the approach are its expressiveness, its support for specifying, constraining, and validating fragments and fragment types, its situational cataloguing abilities, and its accessibility to both software and method engineers.

Keywords: Method Engineering, Situational Method Engineering, Meta-modeling, UML

1 Introduction

A development methodology provides a collection of procedures, techniques, tools, and documentation aids for helping developers in their efforts to implement a system correctly, on time, and within budget [1]. Although sticking to an individual methodology has potential advantages, such as reducing learning and training times and improving the expertise of developers in the chosen methodology, there is no single methodology that can be uniquely pointed as “the best". Hence, different types of "local" adaptations and modifications have to be made in order to adjust a methodology to the specific requirements and constraints of a project. Two areas have emerged for creating and maintaining methodologies: method engineering [2, 18] aims at providing effective solutions for building, improving, and supporting evolution of development methodologies, while Situational Method Engineering (SME) [9, 13] mainly deals with customizing and tailoring methodologies to a specific situation (case). These approaches refer to fragments, the building blocks of methodologies, rather than to complete methodologies. They offer ways to represent

fragments, catalogue them according to different features, retrieve the most appropriate ones, and customize and tailor them to complete methodologies. The quality of fragment representation and cataloguing may significantly affect the quality of the reusing and assembling processes and consequently the quality of the resultant situational methodologies. Hence, these activities are essential and are the focus of this paper, which presents a domain engineering-based approach that allows expressing a large variety of fragments and fragment types, constraining the structure and behavior of fragments, and specifying situational cataloguing information that changes according to the fragment type. These are done using well-know modeling languages and techniques, increasing its accessibility to various users and potentially enhancing user involvement and commitment to the resultant situational methodologies.

The structure of the rest of the paper is as follows. Section 2 introduces and exemplifies the approach, while Section 3 presents its supporting CASE tool. Section 4 analyzes the proposed fragment representation approach in the light of other relevant works. Finally, Section 5 concludes and refers to future research plans.

2 A domain engineering-based approach for fragment representation

The proposed domain-engineering approach is a holistic, visual approach for managing, representing, retrieving, customizing, and tailoring method fragments in order to create new methodologies that best suit a situation at hand. Its fragment representation part provides the ability to express different types of methodologies and their fragments, their associated characteristics and values, their pre- and post-conditions, and other fragment-related requirements, such as mandatory participants, recommended (optional) participants, triggers, etc. We apply the Application-based DOmain Modeling (ADOM) [14, 17] and the standard notation of UML 2.0 [11] for these purposes.

As opposed to other domain engineering [4] and product line software engineering [12] approaches, which are concerned with developing separate techniques and tools for building reusable assets and components that fit to families of applications, ADOM perceives that applications and domains are similar in many aspects, thus it enables modeling domains with regular software engineering techniques. ADOM is based on a three layered architecture: application, domain, and language. The application layer consists of models of particular applications, including their structure and behavior. The language layer includes meta-models of modeling languages, such as UML. The intermediate domain layer consists of specifications of various domains (i.e., application families). These specifications describe the commonality as well as the variability allowed among applications in the domain. The application models use domain models mainly for creation (instantiation, reuse) and validation purposes.

ADOM is a quite general architecture and can be applied to different modeling languages, but when adopting ADOM with a specific modeling language, this language is used for both application and domain layers, easing the task of application

validation by employing the same constructs in both application and domain layers. ADOM-UML, in which ADOM is used in combination with UML 2.0 [11], was chosen in the context of this research due to the familiarity and establishment of UML in the software engineering area. Section 2.1 briefly reviews the principles of ADOM-UML, more elaborated in [14, 17], while the rest of the section focuses on its applicability for representing, cataloguing, and tailoring method fragments.

2.1 ADOM-UML

In ADOM-UML, UML stereotypes are used both for classifying application elements according to their relevant domain elements and for specifying the allowed variability among applications in the domain.
In the language layer, a new <<multiplicity>> stereotype is defined in order to represent how many times a model element of this type can appear in a specific context. This stereotype has two associated tagged values, min and max, which respectively define the lowest and upper most multiplicity boundaries. For clarity purposes, four commonly used multiplicity groups are defined on top of this stereotype, as summaries in Table 1.

Table 1. Defined stereotypes in the language layer of ADOM-UML

Abbreviated notation	Full notation	Meaning
<<optional many>>	<<multiplicity min = 0 max = ∞>>	Any number (including 0) of application elements can be classified (stereotyped) as this domain element
<<optional single>>	<<multiplicity min = 0 max = 1>>	At most one application element can be classified (stereotyped) as this domain element
<<mandatory many>>	<<multiplicity min = 1 max = ∞>>	At least one application element can be classified (stereotyped) as this domain element
<<mandatory single>>	<<multiplicity min = 1 max = 1>>	Exactly one application element can be classified (stereotyped) as this domain element
	<<multiplicity min = n max = m>>	Between n to m application elements can be classified (stereotyped) as this domain element

In the domain layer, the main concepts of the domain and the relations among them are specified using UML. The allowed variability within the domain is also specified in this layer by attaching multiplicity stereotypes to the various domain concepts and by employing the Object Constraint Language (OCL) [10] (e.g., "or" constraints can be used to denote variations and "exclusive or" constraints can be used to denote alternatives).

In the application layer, the stereotype mechanism is used in order to classify the application elements according to the pre-defined domain elements. The classified application elements are required to fulfill the constraints induced by their classifying domain elements at the domain layer. In addition, the ADOM approach allows adding to application models non-classified elements which are specific to the application at hand and, hence, do not appear in the domain model. These additions are allowed as long as they do not violate the domain constraints.

2.2 Representing and cataloguing fragments in ADOM-UML

The structure and guidelines of fragments are described within the domain layer of ADOM, while their instantiations, which specify particular situational methodologies, are defined in the application layer. In these two layers, structural methodological parts, terms product fragments, are described by UML class diagrams, while behavioral methodological parts, termed process fragments, are described by UML activity diagrams. Furthermore, the different features that characterize each fragment are represented and associated to the fragment models as UML templates, which are parameterized elements that can be used to generate other model elements using binding relationships. The exact lists of features that characterize the different types of fragments can be derived from works that have been done in the area of SME, such as [7, 8], and from practitioners.

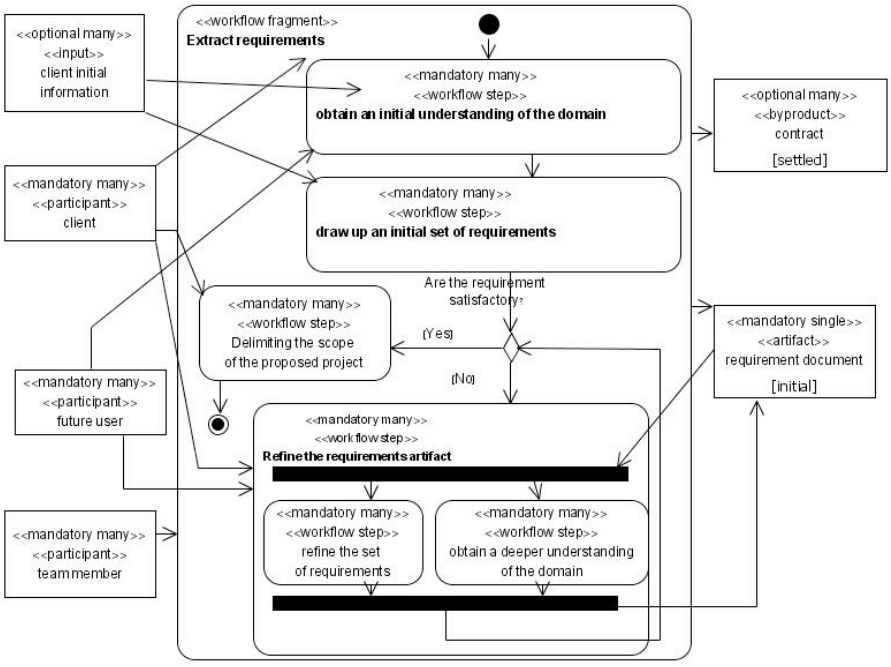

Figure 1. A description of the "extract requirements" process fragment of RUP in ADOM

Figures 1 and 2 respectively exemplify process and product fragments taken from the Rational Unified Process (RUP) [6]. Figure 1 describes the "extracting requirements" workflow, including its optional inputs, required participants, expected deliverables, skeletal steps and flow of control. Figure 2 specifies a "requirement document", which is a special type of product fragments, called artifacts, constraining its general structure and possible (allowed) variability. A requirement document, for example, may relate to several business models and business domain glossaries, which are also artifacts. Note that although constrained, ADOM in general and ADOM-UML in particular allows additional organization-specific features (e.g., attributes or relations) in the requirement classes as long as they do not violate the domain constraints. Figure 2 also specifies the situations in which the usage of requirement documents is desirable: the project life cycle is at least one year, the project size is at least two sub systems, and the flexibility to change is low. As the relevant feature list may become very long, the approach enables specifying the cataloguing information in a separate XML file. However, due to space limitations, we will not elaborate on this option here.

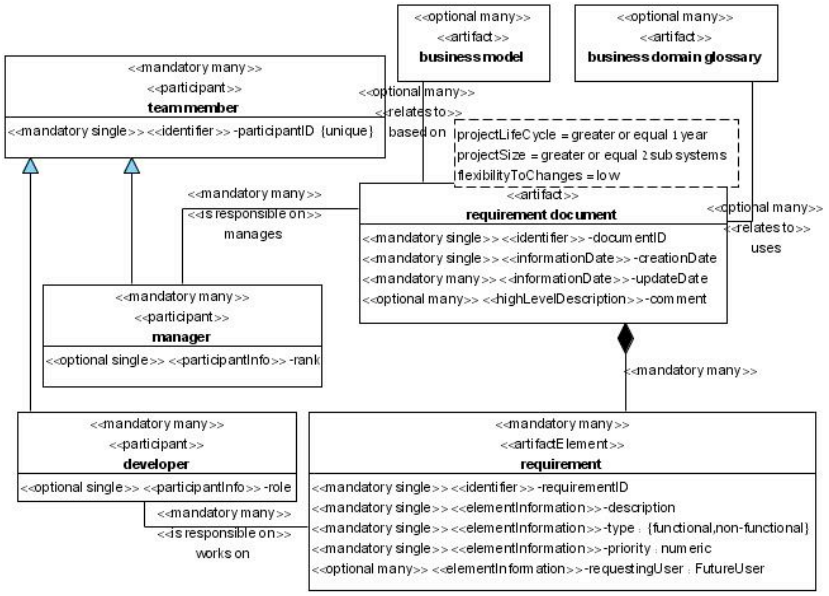

Figure 2. A description of the "requirement document" product fragment of RUP in ADOM

All the stereotypes that are used in these diagrams, except from the multiplicity stereotypes taken from ADOM's language layer, are meaningful concepts in the method engineering area in general and in SME in particular. Hence, they can be (and are) generalized and constrained as more general domain models in ADOM-UML, which capture the knowledge required for specifying particular method fragments in a uniform way. Figure 3, for example, presents a partial model of an artifact. As can be seen, this meta-model is in yet a more abstract level than the fragment models depicted in Figures 1 and 2, allowing its usage for different kinds of artifacts, e.g.,

business models, domain glossaries, and requirement documents. Figure 2 uses the stereotypes defined in Figure 3 and fulfills all the constraints imposed by this figure, including the requirement document attributes and structural relations to other concepts (classes).

2.3 Tailoring methodologies in ADOM-UML

For completeness, we will only demonstrate how the fragment representation and cataloguing activities in ADOM-UML support the consecutive SME processes, or more precisely the tailoring operation. The example shows a simple fragment tailoring which is needed for creating a methodology that is suitable for the Obsert Oglesby case [16].

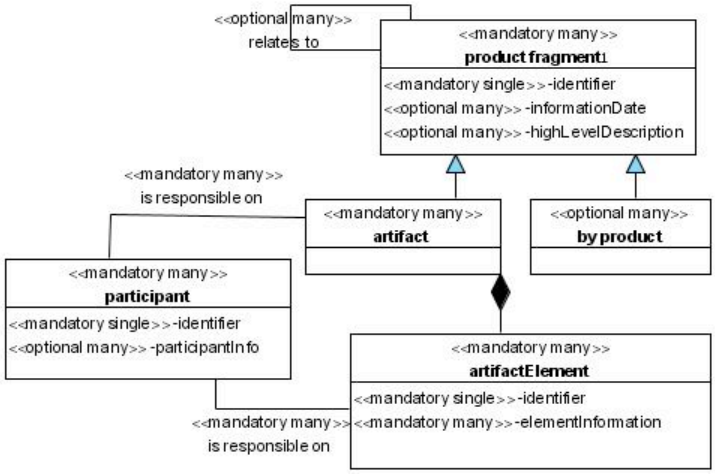

Figure 3. A description of an artifact, which is a specialization of a product fragment

Obsert Oglesby is an art dealer who requests an information system to assist him in buying and selling paintings for his gallery. After consulting with an independent consultant, Obsert decided to turn to a well-known development company to buy a system which will enable him calculating the minimal and maximal prices of a painting and will also serve in detecting new trends in the art market as soon as possible. The development company which was chosen is familiar with the art world and developed similar systems. It mainly works with XP [5] for small projects which need to be developed quickly in an environment of rapidly changing requirements and with RUP [6] for complex projects. Since Obsert's case does not completely fit to any of these options, the development team decided to use suitable fragments from both methodologies and to tailor them for the particular case. The process fragments which were found as relevant to the early development stages of the requested system are "extract requirements" and "build a business model" from RUP and "on-site customer" from XP. These fragments were found by matching the characteristics of the situation (Obsert's case) and his requests to the fragment characteristics. Figure 4 presents a part of the resultant (situational) methodology that tailors the three

retrieved process fragments. The "build a business model" and "extract requirements" fragments are tailored one after the other due to the requirement document, which is an optional input of "build a business model" and the outcome of "extract requirements". The "on-site customer" fragment was tailored in parallel to the two other fragments, due to the absence of common pre- and post-conditions between them. This resultant methodology part belongs to the application layer of ADOM-UML and satisfies all the constraints imposed by the domain models of the composing fragments. The "Obsert requirements elicitation" activity, for example, is an "instantiation" of the "extract requirements" fragment that is detailed in Figure 1. As such, it gets "Obsert Oglesby" as its both client and future user, the "analyst" as its both team member and system analyst, and the "independent consultant report" as its client initial information. It also outputs an "initial requirement comprehension" object as its requirement document and a "settled connection contract" as its contract. However, the internal structure and flow of this "Obsert requirements elicitation" fragment is not shown in Figure 4, as it is irrelevant for tailoring.

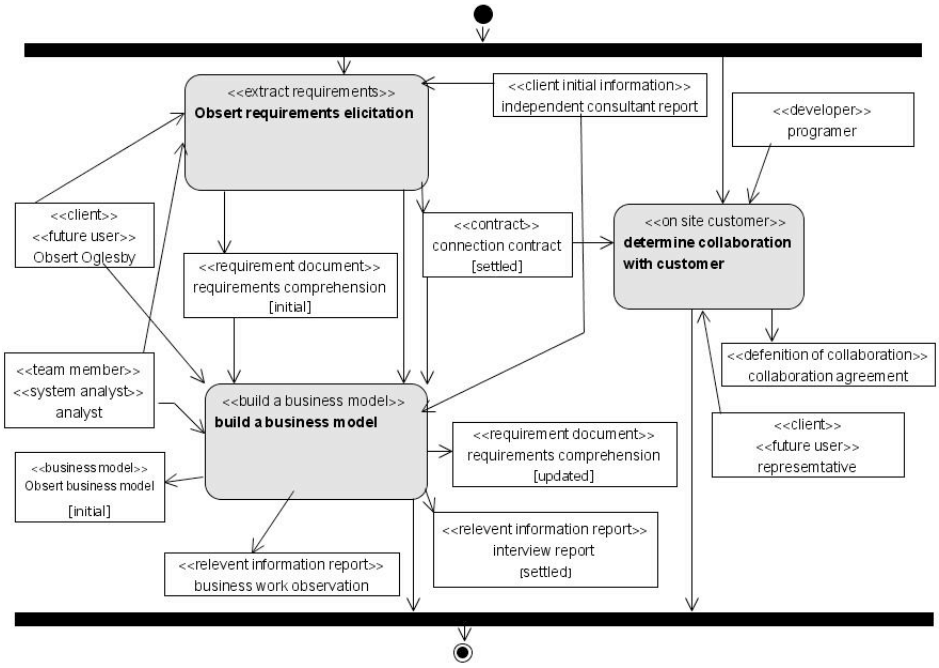

Figure 4. Part of the situational methodology which was created for Obsert's case

3. The supporting CASE tool

Our approach is planned to be accompanied with a CASE tool for managing the aforementioned activities. Figure 5 is an initial design of the main user interface of

this tool. The upper part of the interface gets from the user the situation he/she is facing. It includes sections that refer to the project and organization characteristics, as well as a section for additional constraints, such as the type of methodologies from which the retrieved fragments can be taken. The characteristics list will be dynamically modified, depending on the selected fragment type and previous selections of the user. In the lower left part of the interface, the relevant retrieved fragments are presented. Each fragment is accompanied with a number between 0 and 1 which reflects the distance between the retrieved fragment and the given situation in terms of their exhibited characteristics.

The user will be able to view the different fragments and their characteristics and to select the most appropriate ones. The tool includes a user-friendly editor which basically allows operations supported by UML activity and class diagrams for defining and representing the fragments and their characteristics. It will also support customization and tailoring operations in the context of the new situational methodologies.

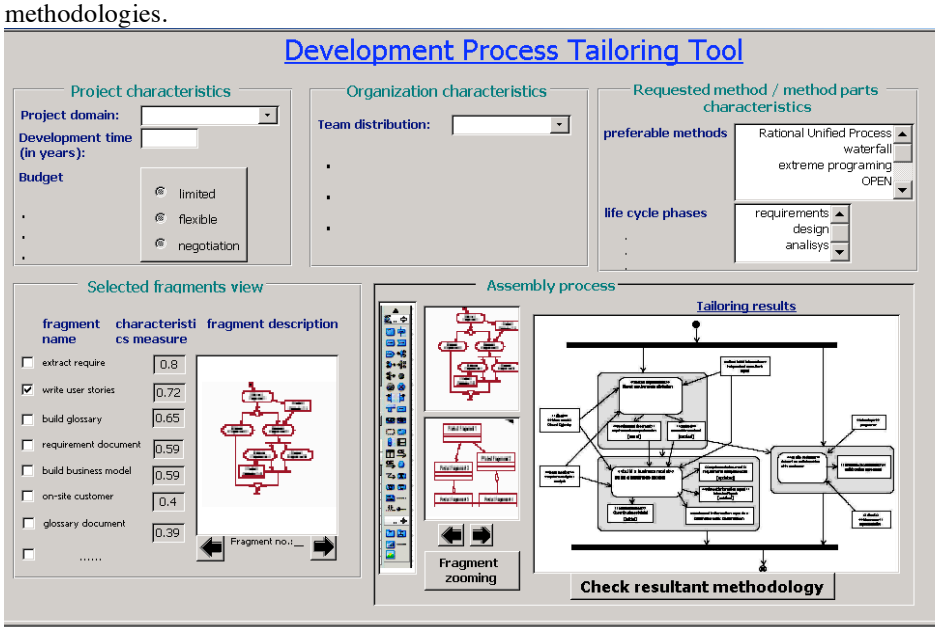

Figure 5. The supporting CASE tool interface

4. Analyzing the proposed ADOM-based approach

The ADOM-based approach represents both process and product fragments in different granularity levels. The abilities to zoom into activities and to decompose classes in UML are employed in order to specify particular fragments to the required level of details without losing the "big picture" of the fragment as a whole. Weerd et al [19] have already proposed class and activity diagrams for supporting web-based

content management system development. However, our approach refers also to fragment types, such as artifacts and workflow fragments, and allows representing them in domain models that capture the relevant knowledge and formally constrain the creation of specific fragments of those types. The particular fragments are required to fulfill the constraints imposed by the relevant fragment types. Furthermore, the usage of fragment types may help integrate, tailor, customize, and assemble particular fragments in a more correct and convenient way.

The separation of fragments into different specifications enables using the same fragment in several contexts, while preserving autonomy of each part. Works which tightly connect product and process fragments into chunks (e.g., [15]) may fall short in (re)using the same fragment in different contexts (e.g., a product that is used by two processes).

In oppose to other works in the area, such as [3] that does not clarify at all (empirically or in other ways) which fragments are suitable and useful for specific situations, the ADOM-based approach supports comprehensive and dynamic definition of organizational, human-related, and project-related characteristics that may be used latter for retrieving and assembling the fragments. These characteristics can be taken, for example, from the reuse frame suggested by [8] which aggregates various works made on methodology aspects. However, since the lists of relevant characteristics may vary between one fragment type to another, our approach allows associating different characteristics to each fragment type, while a specific fragment gets only the relevant characteristics according to its type. Other works in the area, such as [7] and [15], use static, pre-defined lists of characteristics that may be used for all fragments, making the approaches concentrate on only few important characteristics or reducing the ability to control and maintain these lists in the context of a specific fragment.

We chose to specifically use UML in our approach, although the approach can be applied to other modeling languages as well, due to its accessibility to different types of users, including software engineers and managers with technical background. This way we hope to increase the probability of using the resultant situational methodologies and to make the processes of learning and using the fragment representation method easy.

5. Conclusions and future work

As there is no (and probably will not be) a single universally applicable methodology, the importance of SME in general and fragment representation approaches in particular has been increased. We introduced an ADOM-based approach, whose roots are in domain engineering, in order to overcome on some of the main drawbacks of existing fragment representation approaches. In particular, the suggested approach helps identify a wide variety of fragments and fragment types in a uniform way; it supports comprehensive and dynamic definition of cataloguing characteristics; it guides and validates the creation of different types of fragments; and it is accessible to different potential stakeholders, such as software engineers, and not just to method engineers.

As for the future, we plan to define evaluation criteria for all SME activities, specify how the ADOM-based approach supports them in a semi-automatic manner, and compare it to other method engineering and SME approaches. We will also completely develop the supporting CASE tool and examine its usage in industrial companies.

References

1. Avison, D., Fitzgerald, G. Information systems development: Methodologies, Technique and Tools. Second Edition, Higher Education, 2002.
2. Brinkkemper, S. Method Engineering: Engineering of information systems development methods and tools. Information and Software Technology, 38(4), pp. 275-280, 1996.
3. Brinkkemper, S., Saeki, M., Harmsen, F. Assembly techniques for method engineering, CAiSE'98, Lecture Notes in Computer Science 1413, pp. 381-400, 1998
4. Carnegie Mellon Software Engineering Institute. Domain Engineering: A Model-Based Approach, http://www.sei.cmu.edu/domain-engineering, 2002.
5. Extreme Programming Web Site, http://www.extremeprogramming.org, 2006.
6. IBM Web Site, Rational Unified Process, http://www-306.ibm.com/software/awdtools/rup/, 2007.
7. Mirbel, I. Rethinking ISD methods: Fitting project team members profiles. I3S technical report I3S/RR-2004-13-FR, 2004. Available from http://www.i3s.unice.fr/~mirbel/publis/im-isd-04.pdf, 2007.
8. Mirbel, I., Method chunk federation. Available at http://www.i3s.unice.fr/~mh/RR/2006/RR-06.04-I.MIRBEL.pdf, 2006.
9. Mirbel, I., Ralyté, J. Situational method engineering: combining assembly-based and roadmap-driven approaches. Requirements Engineering 11(1), pp. 58-78, 2006.
10. OMG, "Object Constraint Language", Version 2.0, 2005, http://www.omg.org/docs/formal/06-05-01.pdf.
11. OMG, "Unified Modeling Language: Superstructure", Version 2.0, 2005, http://www.omg.org/docs/formal/05-07-04.pdf
12. Pohl, K., Böckle, G. and van der Linden, F. J. Software Product Line Engineering: Foundations, Principles and Techniques. Springer. 1st Edition, 2005.
13. Ralyté, J., Deneckere, R., Rolland, C., Towards a generic model for situational method engineering, CAiSE 2003, LNCS 2681, Springer, pp. 95-110, 2003.
14. Reinhartz-Berger, I., Sturm, A. Behavioral Domain Analysis – The Application-based Domain Modeling Approach, UML'2004, LNCS 3273, Springer, pp. 410-424, 2004.
15. Rolland, C., Plihon, V., Ralyté, J., Specifying the reuse context of scenario method chunks, Proceedings of the 10th International Conference on Advanced Information Systems Engineering (CAiSE'98), LNCS 1413, Springer, pp. 191, 1998.
16. Schach, S. R. An Introduction to Object-Oriented Analysis and Design with UML and the Unified Process. McGraw-Hill/Irwin, pp. 56, 2004.
17. Sturm, A., Reinhartz-Berger, I., Applying the Application-based Domain Modeling Approach to UML Structural Views, ER'2004, LNCS 3288, Springer, pp. 766-779, 2004.
18. Tolvanen, J. P. Incremental Method Engineering with Modeling Tools: Theoretical Principles and Empirical Evidence. Available at http://www.cs.jyu.fi/~jpt/doc/thesis/ime.html, 1998.
19. Weerd, I, Brinkkemper, S., Souer, J., Versendaal, J. A situational implementation method for web-based content management system-application: method engineering and validation in practice. Software process: improvement and practice 11(5): 521-538, 2006.