Enacting a Requirement Engineering Process with Meta-Tools: an Exploratory Project

Sana Damak Mallouli
Centre de Recherche en Informatique
University of Paris 1 Pantheon Sorbonne
Paris, France
sana.mallouli@gmail.com

Saïd Assar
Institut Mines-Telecom,
Telecom Ecole de Management
Evry, France
said.assar@it-sudparis.eu

***Abstract*— An engineering process can hardly be described rigorously because of its dynamic and decision-oriented nature. Applying goal-oriented modeling for representing such processes is a promising approach. However, goal-oriented process models have no clear operational semantics, and building an enactment engine for such models is a highly complex task. To avoid building such tools in an ad-hoc manner and to overcome maintainability and portability issues, we investigate in this paper the practical feasibility of meta-CASE and CAME based approaches for constructing an enactment tool for a goal-oriented model. We describe and analyze a project in which MetaEdit+, a leading meta-tool, is tested. Our aim is to evaluate the meta-tool approach and to explore its possibilities in terms of process model enactment. During this project, a requirement elicitation process is used as a preliminary test bed.**

Keywords-meta-modeling; intentional process modeling; meta-CASE; MetaEdit+; execution semantics.

I. INTRODUCTION

Goal modeling is a prominent design paradigm in various domains such as business process modeling [1], method engineering [2], and requirements engineering [3]. In the method engineering context, the highly dynamic and decision-oriented nature of engineering processes has led to describe such processes using goal-oriented models [4]. The main advantage of goal-oriented process modeling is its ability to go beyond the simple modeling of sequences of activities as proposed by other notations such as BPMN (Business Process Model and Notation) or SPEM (Systems Process Engineering Meta-model). However, how to enact goal-oriented models and how such enactment can be specified and implemented are still open research questions.

The general topic of this paper is how to construct a tool that provides enactment mechanisms for goal-oriented process models. In previous research projects, our research team has engineered multiple tools for earlier event-oriented methods [5], context-oriented [6], and goal-oriented methodological processes [7]. As these tools were built in an ad-hoc manner, recurrent problems have shown up. Maintainability is a key issue: as product and process meta-models are hard coded, it is very difficult to make the tools evolve when the meta-models are changed, even slightly. Portability is a bottleneck too: any evolution of the underlying technology makes the tool rapidly obsolete unless large updates are made to the code. Therefore, we have decided to investigate the possibility of using method engineering tools to solve these issues.

Meta-CASE and Computer Aided Method Engineering (CAME) technologies were introduced in the 90's as an answer to the general problem of providing software support to method and tool customization and/or creation [8]. These meta-tools were expected to facilitate and to accelerate the production of method toolset, and to overcome maintainability and portability drawbacks. As any other software artifacts, tools and meta-tools need to be assessed and evaluated [9]. However, research works that review meta-CASE/CAME tools are rather limited. Prominent evaluations are early works by Martiin & al. [10][11]. Although more recent, works in [12][13] are not fundamentally different from earlier studies. In [12], the evaluation framework is inspired by Martiin & al. [10], and the results are limited to general appreciations. The study in [13] is more detailed, but some studied tools are technically not available anymore (e.g., Mentor [6]). The novelty lies in the evaluation framework; it is based on ISO 9126 quality model with more usability and portability concerns.

What emerge from these previous studies is that when it comes to process enactment, there is clearly not enough empirical knowledge about the use of meta-tool in real cases. Authors claim that support for process modeling is insufficient, and that mechanisms for expressing process enactment are limited or inexistent. So the goal of this paper is to experiment meta-CASE technology for enacting a goal-oriented process, and to explore problems and issues related with such an approach. More precisely, this paper deals with the following research question: To which extent do meta-tools provide a viable and satisfying solution to the problem of building a maintainable and portable enactment engine for a goal-oriented formalism?

To answer this question, we empirically evaluate through a lab project MetaEdit+, a well-known meta-tool. This meta-tool was selected to run the project because it is recognized as a leading method engineering environment with high level of technical maturity [13][14]. The project consisted in using MetaEdit+ to define a goal-oriented formalism and to construct an enactment tool to support it. Evaluation relied on inspecting the obtained system at the end of the project, and on analyzing data gathered along the engineering process. The evaluation is based on a quality framework that

 ISBN: 978-1-61208-283-7

is synthesized from related works analysis. The framework is organized around three perspectives (Table 1): the proposed *formalism* for expressing the process enactment semantics, the *meta-tools* available for specifying the process, and the obtained *target tool* for process enactment.

TABLE I. META-CASE EVALUATION FRAMEWORK

Perspective	Criteria
Formalism	▪ Level of **expressiveness** for process specification ▪ **Cognitive effort** to specify the process part
Meta-tool	▪ **Suitability** for process specification ▪ **Ease of use** for process specification
Target tool	▪ Level of **maintainability** gains for the target meta tool ▪ Level of **portability** gains for the target meta tool

The rest of the paper is structured as follows. In Section 2, we briefly present works that are related to our research. Section 3 describes the Map formalism on which is based the project. In Section 4, the project itself is described. Section 5 presents the outcomes of the project in terms of meta-models and tool specifications, describes the obtained target tool and illustrates with a requirements elicitation process taken from the literature. Section 6 reports our evaluation of the project's outcomes using the quality framework. The paper ends with concluding remarks in which we discuss the results in regard with the research question, and we formulate revised research questions.

II. RELATED WORK

Providing process specification and enactment support is very important for Software and Method engineering fields. In this context, the Moskitt4ME approach [23] is a relevant solution where process models are initially expressed using SPEM meta-model, then transformed into BPMN models. The BPMN process description can then be enacted using an off-the-shelf execution engine called "Activiti Engine".

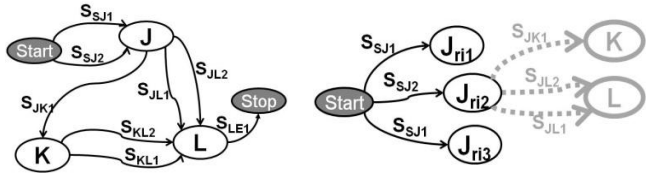
Our research question is similar to the issue addressed in Moskitt4ME approach. However, our solution has two main particularities: (1) first, we use a goal-oriented process modeling language to capture fine grained software engineering processes and to go beyond what SPEM based approaches can do; (2) second, our target is to design a CAME tool itself and to adopt meta-models driven development in order to guarantee maintainability and portability.

Another recent work dealing with enactment process is presented in [24]. This work proposes xSPEM tools (i.e., executable SPEM) for editing, simulating and verifying SPEM process models. The approach is based on dynamic meta-modeling (i.e., the definition of domain specific languages with behavioral semantics) in order to simulate and validate models. The basic idea is to extend the meta-model and assimilate its execution semantics to that of state-based machines and workflows.

xSPEM allows the simple modeling of sequences of activities. Thus, it is not well suited to represent engineering processes. For this reason, we are investigating goal-oriented process enactment in this paper.

III. THE MAP FORMALISM

Our research work concerns the Map formalism, a goal-oriented model that is particularly well adapted for representing engineering processes. Based on the intention paradigm [15], the Map captures the intentions that a process is expected to fulfill, together with a set of available strategies to realize these intentions (Fig. 1a). Each intention can be realized by one or more strategy, and the process is represented as a labeled graph with intentions as nodes and strategies as edges [16]. An edge enters a node if its strategy can be used to achieve the intention of the node. A section of the Map is a triplet composed of a source intention, a target intention and a strategy (e.g., <J, S_{JK1}, K> in Fig. 1a).



(a) (b)

Figure 1. A map example (a), and an illustration of achieved intentions and candidate sections (b)

Map operational semantics. Beyond precedence relationships between intention achievements, the operational semantics of the Map are decision-oriented. The combination of a past intention achievement, a strategy and an intention (i.e., the triplet <Jri_2, SJL_1, L> in Fig. 1b) is called a *candidate section* and is a fundamental concept for expressing the operational semantics. At each execution of a section, a new set of candidate sections (i.e., sections that can be executed in the next step) is computed. Given a certain state of the working products (i.e., the set of product instances,) and the history of achieved intentions, the candidate sections computation is done by checking which sections match with scheduling possibilities [17].

IV. PROJECT OVERVIEW

The chosen meta-tool for the project is MetaEdit+ [18]. Figure 2 presents an overview of the project's main steps.

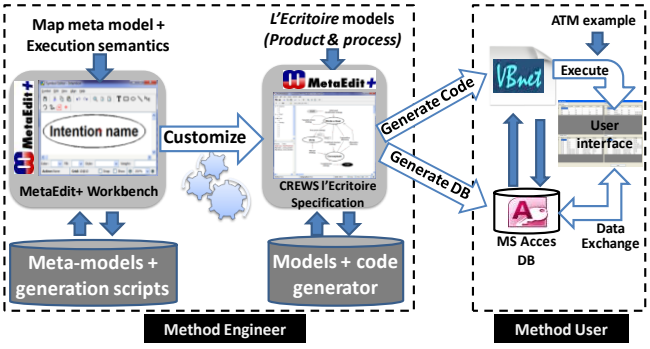


Figure 2. Overview of the engineering process underlying the exploratory project

The method engineer defines meta-models (product and process) using MetaEdit+ Workbench toolset and the

GOPRR (Graph, Object, Property, Relationship, Role) meta-modeling concepts, and specifies generation rules with Merl scripting language. A customized version of MetaEdit+ editor is automatically constructed (i.e., target tool) this tool includes code generation functionalities. The method user can then use this tool to define a product and a process model, and generates an enactment engine.

The project was run with the requirement elicitation process underlying CREWS *L'Ecritoire* [19][7]. Goal discovery and scenario authoring are complementary activities with CREWS *L'Ecritoire*: once a goal is discovered, a scenario is authored as a possible concretization of the goal, and can then be followed by further goal discovery from the authored scenario.

These goal-discovery/scenario-authoring sequences are repeated to incrementally populate the requirement chunks hierarchy (a *Requirement Chunk* (RC) is defined as a <goal, scenario> couple). The underlying RC elicitation process is described as a map.

This project was conducted by a group of three persons: a senior researcher, a PhD student and a master student. The senior researcher controlled the project, collected data and provided support all along the project in case of specific problems. The PhD student defined the Map formalism with MetaEdit+ and provided assistance in expressing and validating Map operational semantics. The tasks of the MSc student were to specify the code generator in MetaEdit+ scripting language for the target platform (VB.NET and MS Access), and to run CREWS *L'Ecritoire* case example.

V. PROJECT OUTPUT

A. Meta-Modeling

For the product part, as the Map formalism does not specify product models, we defined a standard E/R style product meta-model. This meta-model is simple and is not presented here for the sake of space. Figure 3 presents the meta-model for the process part expressed in GOPRR. Boxes represent objects, diamonds represent relationships, and circles are roles. *Intention* and *Strategy* concepts are defined as objects. They are connected by two relationships (*Strategy-to-Intention* and *Intention-to-Strategy*) and, beyond name and identifier, have certain properties necessary for expressing operational semantics (i.e., "State" attributes). "Product fragment", "Product consumption" and "Product generation" are specific attributes which are added to the Map definition. They are necessary for expressing the behavior of a map when it is executed [20], and play an important role in computing candidate sections.

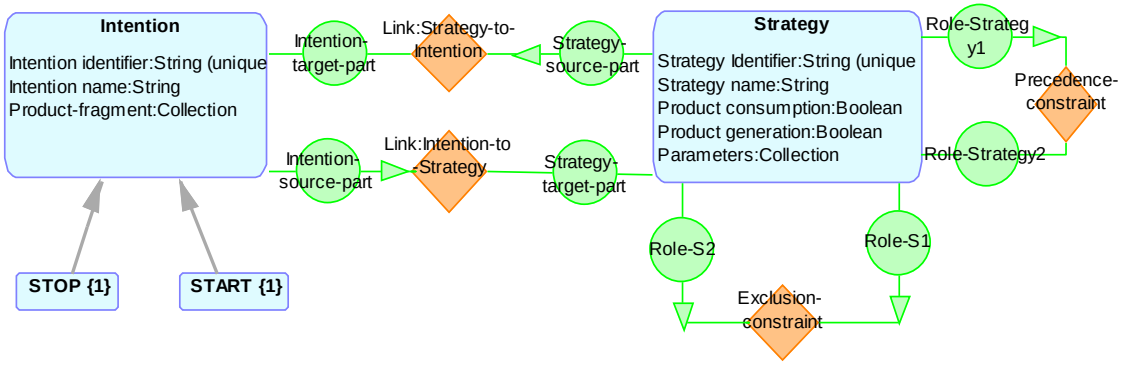


Figure 3. Map process meta-model defined in GOPRR

B. Target Tool for CREWS L'Ecritoire

The product model is simple; it contains three objects (i.e., a requirement chunk, a goal and a scenario) and three links (And, Or, Refine). For the sake of space, details are not presented here, but interested readers can refer to [16].

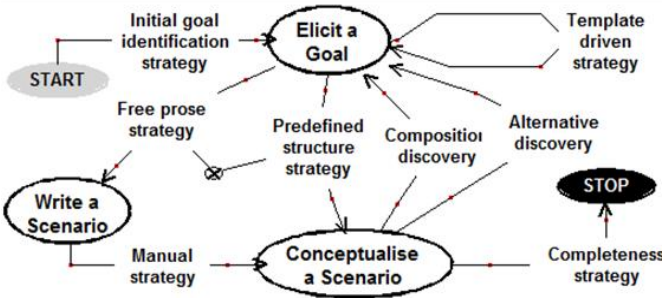


Figure 4. CREWS L'Ecritoire process model defined in the target tool

Figure 4 presents the requirement chunks elicitation process expressed as a map. Three intentions are defined together with a set of strategies to achieve them; this model is a slightly simplified version of the original process in [16].

C. Enactment specification

Enactment of a specific map is handled by the generated Map engine. This engine is a VB.NET program which manipulates an MS Access DB. The executable code of the Map engine and the relational structure of the DB are product dependant, and will be constructed by the code generator according to the product model (i.e., requirement chunks) and to the process model (i.e., the map). Indeed, for each method, a specific Map engine is generated; however, the code generator is generic.

Figure 5 shows the structure of the engine's DB that is generated for the CREWS *L'Ecritoire* case. Tables in the "Process" group are generic and contain the map, those in the "Trace" group will contain the results of the execution

(product dependant), and those in the "Product" group will contain product instances (i.e., requirement chunks).

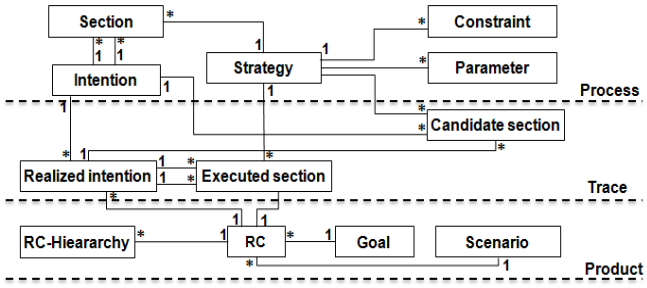


Figure 5. Database structure of the Map engine for CREWS *L'Ecritoire*

The dependency of the Map engine towards the product structure impacts also the generated algorithm for candidate sections computing, which is an essential element in the enactment process. This calculation is based on the content of the engine's DB. For each realized intention Ri in the trace, a set of candidate sections is calculated by linking Ri to the corresponding intention I in the process map, and identifying all connecting strategies and connected intentions which form a possible path from Ri.

Figure 6 presents a screen shot of Map engine execution for CREWS *L'Ecritoire* case. From left to right top down, the 1st window contains the list of executed sections, the 2nd contains the stack of achieved intentions, the content in the 3rd window is static as it contains the sections of the process map under execution, and the last window contains the dynamically computed list of candidate sections.

VI. Evaluation and Discussion

The quality framework (cf. Table 1) will now be used to evaluate and discuss the project results and outcomes. Results are synthesized in Table 2.

A. Formalism and meta-tool

For the process part, suitability and usability are clearly insufficient. Operational semantics are specified through a collection of scripts written in Merl (MetaEdit+ scripting language). These scripts define navigation logic through the meta-models and output (i.e., generate) instruction in target code (i.e., VB.NET). This step is very complex, and must be handled in two sub steps: write, run and debug first the target code for a sample Map engine DB, and when this code is satisfying, adapt it and integrate it into Merl generation scripts. Errors in the generated code are very hard to be directly corrected in the generating Merl scripts. Although the meta-tool provides a debugging facility, it is largely insufficient when the generated code becomes complex. The cognitive effort for correlating the generated code with procedures for generating such code (i.e., Merl scripts writing) was so high that the whole project team needed to cooperate frequently on this task.

B. Target tool

According to the project team perception, the obtained Map engine operates correctly and is easy to use, as the possibilities for interactions are limited to choosing a candidate section and inputting some data. However, the user interface is much less attractive although the Map enactment is made sufficiently clear through textual display windows (Fig. 6). This can be enhanced; it will however make the generated code and the generation scripts much more complex and less portable.

Concerning target tool maintainability, the scoring is mitigated. Any change to the Map formalism which does not impact the operational semantics is easily handled with MetaEdit+'s graphical interface and with small updates to the generated code. However, if the modification changes the operational semantics, the impact on the generated code can then be much larger.

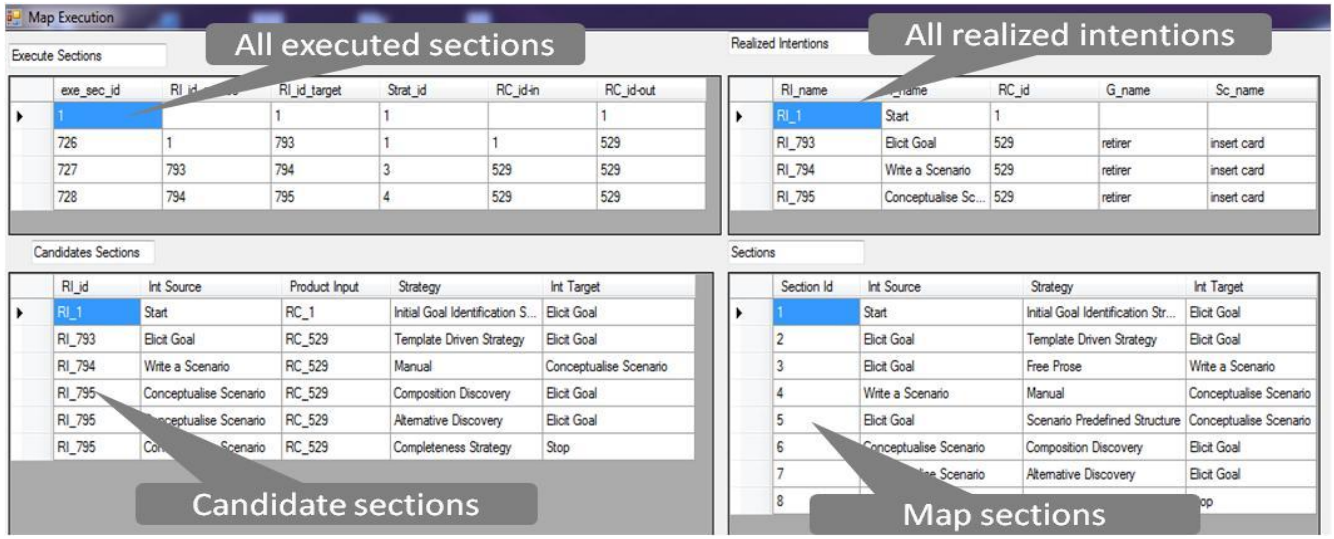


Figure 6. Interface for the generated target tool

The portability issue is strategically more important as it impacts tool mid and long term existence. The scoring here is unsatisfactory because, although it is fully possible to update the generating scripts and the generated code to accommodate a new version of the target platform (e.g., a new version of VB) or a different platform (i.e., MySQL DBMS), the cost of this task is high. However, compared with an ad-hoc approach, the project is insufficient to get a real insight about the comparative advantage of using meta-CASE in terms of portability.

TABLE II. PROJECT EVALUATION RESULTS

Perspective	Criteria	Evaluation result
Formalism	Expressiveness	▪ Insufficient: execution semantics is expressed using a large set of procedural scripts
	Cognitive effort	▪ Very high: the language designer must mentally correlate code generation actions (i.e., Merl scripts) with generated code (i.e., VB.net code)
Meta-tool	Suitability	▪ Low: a simple text editor for developing generation scripts, there is no model nor a GUI
	Ease of use	▪ Low: the validation of a script is very hard, with only a limited debugging facility
Target tool	Maintainability	▪ Satisfactory in case of modifications to the PLM that do not alter its execution semantics; unsatisfactory otherwise, because of complex changes to code generation scripts
	Portability	▪ Unsatisfactory: moving to a different target platform induces high level of changes to code generation scripts

VII. CONCLUDING REMARKS

In this paper, we have reported an experimental development project dedicated to evaluating the contribution of meta-CASE to the enactment of Map, a goal-oriented process modeling formalism. As "building a system in and of itself does not constitute research", the contribution to basic research is "the synthesis of new concepts in a tangible product" [21]. What we learned from this experience is resumed in the two following. First, providing mechanisms for enacting a goal-oriented process is possible only if operational semantics can – at the conceptual level – be expressed using the execution paradigm underlying a certain target platform. This is line with recent work about the necessity to raise the level of abstraction of compilation techniques up to the modeling phase [22]. Second, in the Map case, building a generic Map engine is impossible as the algorithm for candidate section computation is product dependant. Meta-CASE technology provides thus an interesting possibility to overcome this problem by combining CASE customization and the design of adequate code generators. This opens the door to multiple applications in the field of Domain Specific Languages, CASE tool construction and process enactment.

A. Threats to validity

Some threats to the validity of this study are: the limitation to only one project, the dependence on participants' background, and the mix between development team and observation team.

B. Research question revisited

In meta-CASE actual technology, process operational semantics are expressed in a procedural manner in code generating scripts, and process enactment is obtained through the execution of generated code. These specifications are loosely correlated with the meta-modeling part, they are not specified in a declarative manner and they cannot be represented graphically. They are difficult to build and to validate. Indeed, like in compiler technology, the validity of the code generator cannot be demonstrated unless some formal notations are used, or the code generator itself is built using some generic tool. Thus, we revise our research question into: How to express in a rigorous manner process operational semantics at the meta-modeling level of abstraction in order to facilitate its validation and evolution? This is the subject of our actual research work. We are investigating graphical notations for expressing meta-models' operational semantics.

REFERENCES

[1] I. Bider and P. Johannesson, "Goal-oriented business process modeling: Guest Editorial," in Business Process Management Journal, vol. 10, no. 6, 2005, pp. 621–623.

[2] J. Ralyté and C. Rolland, "An Approach for Method Reengineering," in H. S.Kunii, S. Jajodia, and A. Sølvberg, "Conceptual Modeling," ER 2001, Berlin/Heidelberg, LNCS, Springer, vol. 2224, 2001, pp. 471–484.

[3] A. Lapouchnian "Goal-oriented requirements engineering: An overview of the current research," University Of Toronto, Canada, 2005.

[4] S. Si-Said and C. Rolland, "Formalising Guidance for the CREWS Goal-Scenario Approach to Requirements Engineering," in H. Jaakkola,H. Kangassalo, and E. Kawaguchi, "Information Modelling and Knowledge Bases," IOS Press, vol.10, 1999, pp. 172–190.

[5] C. Rolland and al., "The RUBIS system," in T.W. Olle, A.A. Verrijn-Stuart, and L. Bhabuta, "Computerized Assistance During the Information Systems Life Cycle," North-Holland, 1988, pp. 193–239.

[6] S. Si-Said, C. Rolland, and G. Grosz, "MENTOR: A Computer Aided Requirements Engineering Environment," in CAiSE'96, P. Constantopoulos, J. Mylopoulos, and Y. Vassiliou, LNCS, Springer, Berlin/Heidelberg, vol. 1080, 1996, pp. 22–43.

[7] C. Souveyet and M. Tawbi, "Process centred approach for developing tool support of situated methods," in DEXA'98, G. Quirchmayr, E. Schweighofer, and T.J.M. Bench-Capon, LNCS, Springer, Berlin/Heidelberg, vol.1460, 1998, pp. 206–215.

[8] S. Kelly and K. Smolander, "Evolution and issues in metaCASE," Information and Software Technology, vol. 38, no. 4, 1996, pp. 261–266.

[9] J.P. Gray, A. Liu, and L. Scott, "Issues in software engineering tool construction," Information and software technology. 2000, Vol. 42, no. 2, pp. 73–77.

 ISBN: 978-1-61208-283-7

[10] P. Marttiin, M. Rossi, V.-P. Tahvanainen, and K. Lyytinen, "A comparative review of CASE shells: A preliminary framework and research outcomes," in Information & Management, vol. 25, no. 1, 1993, pp. 11–31.

[11] P. Marttiin, F. Harmsen, and M. Rossi, "A functional framework for evaluating method engineering environments: the case of Maestro II/Decamerone and MetaEdit+," in S. Brinkkemper, K. Lyytinen, and R.J. Welke, "Method engineering: Principles of method construction and tool support," IFIP, Chapman & Hall, London, 1996, pp. 63–86.

[12] I. Van de Veerd and M. Saeki, "An evaluation of computerized tools for method construction," Institute of Electronics, Inf. and Comm. Engineers (IEICE), Tokyo, Japan, 2007.

[13] A. Niknafs and R. Ramsin, "Computer-Aided Method Engineering: An Analysis of Existing Environments," in Z. Bellahsène and M. Léonard, CAiSE'08, LNCS, Springer, Berlin/Heidelberg, vol. 5074, 2008, pp. 525–540.

[14] S. Kelly and J.-P. Tolvanen, "Domain-specific modeling: enabling full code generation," Wiley-Interscience, IEEE Computer Society, N.J. Hoboken, 2008.

[15] C. Rolland, "Capturing System Intentionality with Maps," in J. Krogstie, A.L. Opdahl, and S. Brinkkemper, Conceptual Modelling in IS Eng., Springer, 2007, pp. 141–158.

[16] C. Rolland, N. Prakash, and A.Benjamen, "A Multi-Model View of Process Modelling. Requirements Engineering," Requirements Engineering, vol. 4, no. 1, 1999, pp. 169–187.

[17] M.H. Edme, "A proposal for intentional modeling and guiding of information system usage (in French)," PhD thesis, University of Paris 1 La Sorbonne, France, 2005.

[18] MetaCASE: http://www.metacase.com/ [retrieved: April 25th, 2013]

[19] C. Rolland, C. Souveyet, and C.B. Achour, "Guiding goal modeling using scenarios," IEEE Transactions on Software Engineering, vol. 24, no. 12, 1998, pp. 1055–1071.

[20] S. Assar, S.D. Mallouli, and C. Souveyet, "A behavioral perspective in meta-modeling," in 6th Int. Conf. on Software and Data Technologies (ICSOFT), Sevilla, Spain, 2011.

[21] J.F. Nunamaker, M. Chen, and T.D.M. Purdin, "Systems development in information systems research," Journal of Management Information Systems, vol. 7, no. 3, 1990, pp. 89–106.

[22] R. Bendraou, J.M. Jezéquél, and F. Fleurey, "Achieving process modeling and execution through the combination of aspect and model-driven engineering approaches," Journal of Software: Evolution and Process, vol. 24, no. 7, November 2012, pp. 765–781.

[23] M.Cervera, M. Albert, V. Torres, and V. Pelechano, "The MOSKitt4ME Approach: Providing Process Support in a Method Engineering Context," in P. Atzeni, D.Cheung, and S. Ram, Conceptual Modeling (ER), LNCS, Springer Berlin Heidelberg, 2012, pp. 228‑241.

[24] R. Bendraou, B. Combemale, X. Cregut, and M.-P. Gervais, "Definition of an Executable SPEM 2.0," Proc. 14th Asia Pacific Software Engineering Conf. (APSEC), 2007, pp. 390-397.

 ISBN: 978-1-61208-283-7