HAL
archives-ouvertes.fr

A behavioral perspective in meta-modeling

Saïd Assar, Sana Mallouli, Carine Souveyet

▶ **To cite this version:**

Saïd Assar, Sana Mallouli, Carine Souveyet. A behavioral perspective in meta-modeling. ICSOFT 2011 : 6th International Conference on Software and Data Technologies, Volume 2, Jul 2011, Séville, Spain. pp.238-243. hal-01513015

HAL Id: hal-01513015
https://hal.archives-ouvertes.fr/hal-01513015

Submitted on 24 Apr 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A BEHAVIORAL PERSPECTIVE IN META-MODELING

Saïd Assar
Institut Telecom, Telecom Business School, 9, rue C. Fourier 91011 Evry – France
said.assar@it-sudparis.eu

Sana Damak Mallouli, Carine Souveyet
Université Paris 1 La Sorbonne, Centre de Recherche en Informatique
90, rue de Tolbiac, 75013 Paris – France
{sana.mallouli, carine.souveyet }@univ-paris1.fr

Keywords: Meta-modeling, method engineering, model executability, behavioral perspective in meta-modeling, event-based meta-modeling, meta-CASE tool, CAME tool.

Abstract: Meta-models are essential artifacts for specifying and reasoning on models and on methods. Traditionally, meta-modeling follows the "data" perspective and only the structural part of a model is represented. The "process" and "behavior" perspectives are neglected or partly represented, and for a process meta-model, such specifications express its enactment and execution semantics. From a Computer Aided Method Engineering (CAME) point of view, such specifications are necessary for enacting the process part of a method when specified. In this paper, we defend the position that in process meta-modeling, it is essential to include the behavior perspective, and that event-based meta-modeling can help in expressing, graphically and at high level of abstraction, the executable semantics of a process modeling notation. We illustrate this approach through the construction of event-based meta-models for the intention oriented Map notation.

1 INTRODUCTION

A meta-model is a formal specification of a model that helps in understanding it and in reasoning on its structure, its semantics and its usage. Meta-modeling, which is the activity of constructing meta-models, is widely used in Information Systems (IS) engineering and especially in model design and method engineering (Brinkkemper et al., 1996), (Rolland, 2007b). It is a powerful conceptual tool to analyze product and process models, and to design corresponding CASE tools.

In the literature, meta-modeling is generally used to specify meta-models that reflect the static structure of models, i.e. concepts and links between these concepts (Jeusfeld et al., 2009), (Sprinkle et al., 2011). For instance, if we consider the meta-model shown in figure 1, which is an extract of the SPEM meta-model represented in UML, we notice that this specification describes the structural dimension of this process model. It represents SPEM concepts and how they are inter-linked. How these elements interact during the execution is not explicitly expressed in the meta-model.

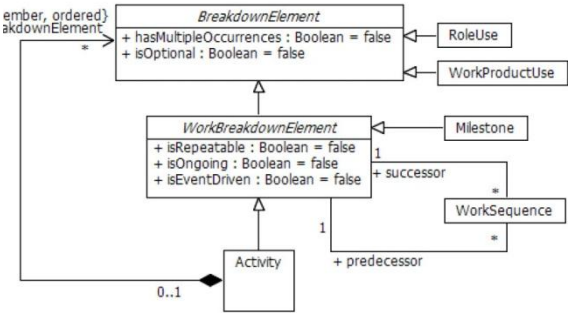

Figure 1: A fragment of the SPEM Meta-Model, extracted from (OMG, 2008), p.54

While the "process" and the "behavior" perspectives are well-known in IS modeling (Olle et al., 1991), they are generally missing in meta-models specified in the software engineering field. Depending on the nature of the studied model, the lack of these perspectives deprives tools designers and method engineers of an important knowledge about the models they are manipulating. In the case of a process meta-model, the "process" and "behavior" perspectives inquire in fact on the executable semantics of the underlying model.

The goal of this paper is to present and discuss how to take into account the "process" and "behavior" perspectives when specifying at the meta-level a process model. We are particularly interested in process models with interactive behavior. Indeed, these models (such as BPMN, Workflow, etc.) were designed to represent organizational systems involving external agents to the system. To express these interactions and the underlying semantics, corresponding meta-models must take into account not only the structural perspective (concepts and relationships), but also dynamic and behavioral perspectives.

This paper is organized into 5 sections. Section 2 presents related works in specifying models executability. Section 3 briefly provides the basics of the meta-modeling notation which will be used. Section 4 is an illustration of our approach applied on the intentional Map model. Section 5 discusses the advantages and disadvantages of the proposed approach, and proceeds with the conclusion.

2 RELATED WORKS

In software engineering, expressing model executability in meta-models has been studied extensively, particularly since MDA (Model Driven Architecture) and MDE (Model Driven Engineering) approaches to software development have been introduced. Indeed, given that the MDE approach is fundamentally based on the extensive use of models at all phases of software development, the question of how to execute a model and how to express its executable semantics quickly arose. A first study on the relationship between a meta-model and the problem of expressing the executability of the underlying model was made on Petri nets in one of Bézivin works (Breton et al., 2001). The authors complement the static meta-model describing the structure of the model (arcs and transitions in a Petri net) by a dynamic meta-model which introduces data structures necessary for the execution of an instance of this model (tags and movement of token). However, the authors acknowledge that this is not sufficient to express the model full executability as the used formalism (UML class diagram) has no executable semantics, and the authors call for the creation of an executable UML. And it is probably the result of these preliminary thoughts on the problem of expressing model executability in meta-models that the Kermeta language was proposed and developed (Muller et al., 2005). Kermeta is an object-oriented meta-programming language with a software environment designed for meta-model engineering. It provides a way to add meta-specification to an UML meta-diagram. The Kermeta meta-programming language has been used to build a comprehensive and executable specification for simple models like Finite State Machine (Kermeta, 2011).

Further works in the software engineering and the MDE communities focused on studying engineering processes models, because of the importance of describing, controlling, and automating procedures by which software systems are constructed. UML4SPM is an important work in this register (Bendraou et al., 2005). It defines a modeling language for representing and enacting engineering process models. It is based on UML, and is similar to the OMG's SPEM standard. Several experiments were made to specify the semantics and express the executability of UML4SPM using the BPEL processes execution language and the meta-specification language Kermeta (Bendraou et al., 2007). For both approaches, the problem of interacting with the system environment (the user or other systems) is highlighted.

In IS engineering domain and especially in method engineering field, few studies to our knowledge have addressed the question of the explicit expression of executability in process meta-model. As a method definition is a combination of product and process meta-models, product meta-model specifications are historically the oldest (Harmsen et al., 1996). In (Brinkkemper et al., 2001), the MEL language, which is a formal language for specifying methods, is proposed. Apart the structural specification of components, the process aspect is described in MEL using formal operators whose semantics is guaranteed by the underlying mathematical notation. This approach by assembling components methodology is currently predominant (Henderson-Sellers et al., 2010); however, there are still no models to formalize the approach, neither to specify the methodological component, nor to formally express the assembly process (Seidita et al., 2007).

To conclude this overview, we have to mention meta-modeling formalisms and languages proposed by CAME and meta-CASE environment. MetaEdit is a well known tool which allows specifying a meta-model using the data-oriented static notation GOPRR (Kelly et al., 1996), and generating a graphical editor for the specified model (Kelly et al., 2008), (MetaCASE, 2011). The "process" and the "behavior" perspectives are relegated to the phase of code generation where instances of the model can be

manipulated and corresponding instructions can be generated in any target language (i.e. XML, C++, Java) using the MERL scripting language. Whereas the meta-model definition is declarative using a graphic interface, executability is expressed in an operational way with a standard programming interface. This is the main drawback of MetaEdit.

ConceptBase is another meta-modeling formalism supported by a meta-CASE environment. which is based on the Telos model (Mylopoulos et al., 1990), and is implemented using the Datalog logic based language. ConceptBase is a powerful graphical meta-modeling environment allowing to specify any number of abstraction levels, and to express constraints and queries on several of these levels. Regarding "process" and "behavior" perspectives, ConceptBase introduced Event-Condition-Action (ECA) rules to express the dynamics of a meta-model. An illustration is given in (Jarke et al., 2010) with the rules of execution of a Petri net.

3 EVENT BASED META-MODELING NOTATION

The aim of this paper is to show the importance of the behavioral perspective by applying it in specifying a process meta-model. We argue that such a description can improve model specification and consequently facilitate the implementation of a corresponding CASE tool. For this purpose, we will first use the UML class diagram to specify the data perspective of the meta-model. This model can be complemented with the UML sequence diagram to express the process perspective. For sake of space, this step will not be shown here. Finally, for specifying the behavior of the model, we introduce an event-based notation directly inspired from the Information System Development Framework (Olle et al., 1991). The behavior perspective is built upon the concepts of event, trigger and operation (figure 2). An event is characterized by its name, its type (internal or external), and a predicate expressing the condition of its occurrence. An *external event* corresponds to the arrival of a message, while an *internal event* is related to a state change in an object. A message is issued by an agent, which can be a human actor or an application system. It is a set of structured data which is relevant and significant for the system. An Agent is described by its name, its type (human or system), a set of incoming messages and a set of out coming messages.

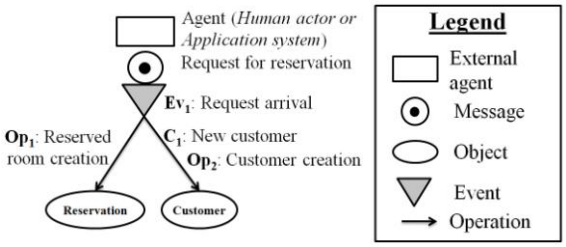

Figure 2. Graphical notation for representing the behavior perspective in meta-models

The *ascertain* relationship is defined either between an event and a message for an external event, or between an event and an object in case of an internal event. The trigger relationship relates an event to one or several operations. A trigger body is composed of a flow of unsorted atomic operations to be executed when the event occurs. This execution can be conditional; in this case, a specific condition is associated to the triggering of the operation.

The main advantages of this notation are its simplicity and the availability of a graphical representation. An important feature is the emphasis on the interaction between the system application tool and the external environment.

4 SPECIFYING THE MAP META-MODEL

A map is a labeled directed graph with intentions as nodes and strategies as edges (Rolland, 2007a). An edge enters a node if its strategy can be used to achieve the intention of the node. Since there can be multiple edges entering a node, the map is capable of representing many strategies that can be used for achieving an intention. A map is a non deterministic representation of a process. We call "Section" a triplet composed of a source intention, a target intention and strategy. The Map formalism do not constrain the user in a sequential process consisting of successive steps, but allows instead a large degree of freedom in the scheduling of intentions and in the choice of the strategy to be applied at each step in the process. The UML class diagram in figure 3 depicts the static structure of the Map meta-model.

The meta-model contains on one hand representations for the Map concepts ("Map", "Intention", "Strategy", "Section", "Constraint", "Situation"), and on the other hand, additional structures necessary for handling the enactment of a map ("Intention_Realisation", "Section_Execution", "Trace", "Map_application", "Map_User").

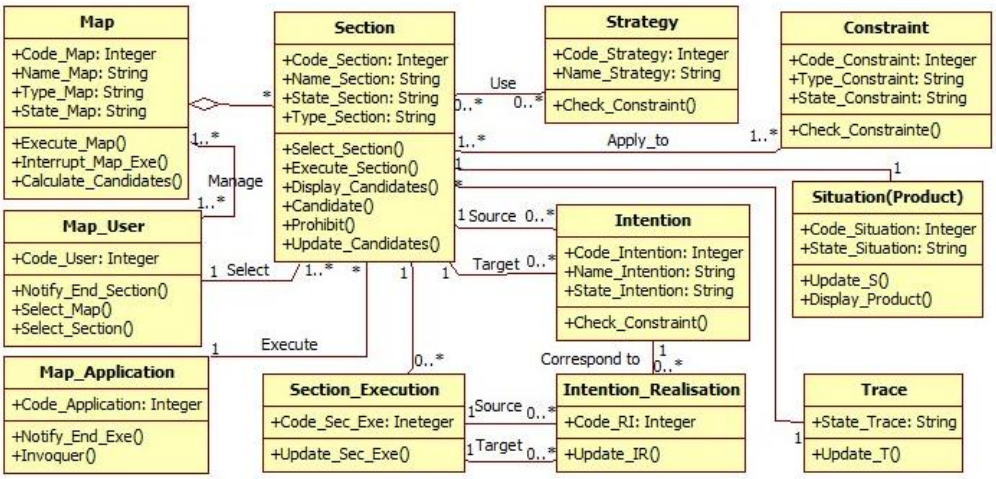

Figure 3: Static representation of the Map meta-model

A map is enacted one section at a time. At each step of the enactment process, a new set of candidate sections (sections that can be executed in the next step) is computed. This is done by checking those sections that respect the scheduling constraints, and that match with the given current situation of the working products ("Situation"), and that match with the given history of the process execution ("Trace"). From this set of candidate section, a section with the corresponding product fragment (i.e. the situation) is interactively selected by the user (Assar et al., 2000).

We have defined in the meta-model additional attributes such as the "state" attribute in several classes to track the current state of an object and its evolution during the enactment process. For example, the attribute 'State_Section' in the class "Section" takes the values ('Selected', 'Executed', 'Candidate', 'Prohibited', 'Running'). This information indicates the changing status of the class "Section" during the execution of a map, and thus plays an important role in reasoning about the progress of the enactment process. Finally, the classes "Map_User" and "Map_Application" represent external agents that interact during the enactment of a map. They contain necessary data about human and software users of the system. Without these elements of information, it is impossible to know when and why an operation will be executed.

The class diagram in figure 3 is a partial representation of a procedural vision of the enactment process. It is insufficient for designing a meta-model-driven Map enactment tool because the way interactions are handled, is not explicitly represented. That's why we propose to use the behavior perspective to describe the causal relationship between the Map enactment tool and environment, together with the inside event driven logic of the enactment process. Figure 4 corresponds to the dynamic schema of the Map meta-model. This graphic representation reflects the systemic view of the Map execution. For the sake of place, only some events will be briefly detailed in this paper (table 1).

Table 1: Specifications for the behavioural meta-model.

EV1	- arrival of an "Execute map" message - triggers the "Execute_Map" operation of the class "Map" - sets the value to 'Selected' in the attribute "State_Map"
EV2	- the value of "State_Map" changes from 'Selected' to 'Running' - triggers the computation of candidate sections
EV3	- the value of attribute "State_Section" changes from 'Prohibited' to 'Candidate' - triggers the display of all candidate sections
EV4	- the user selects an item among the list of candidate sections - modifies the "State_Section" attribute value from 'Candidate' to 'Selected'
EV5	- attribute "State_Section" change its value from 'Candidate' to 'Running' - triggers the execution of the selected section and invokes an external application to perform the task associated with a strategy
EV6	- the execution of a section is finished, "State_Section" changes from 'Running' to 'Executed' - triggers the update of the trace, insert a new realized intention, updates the situation of the product and notifies the user of the end of execution of the selected section

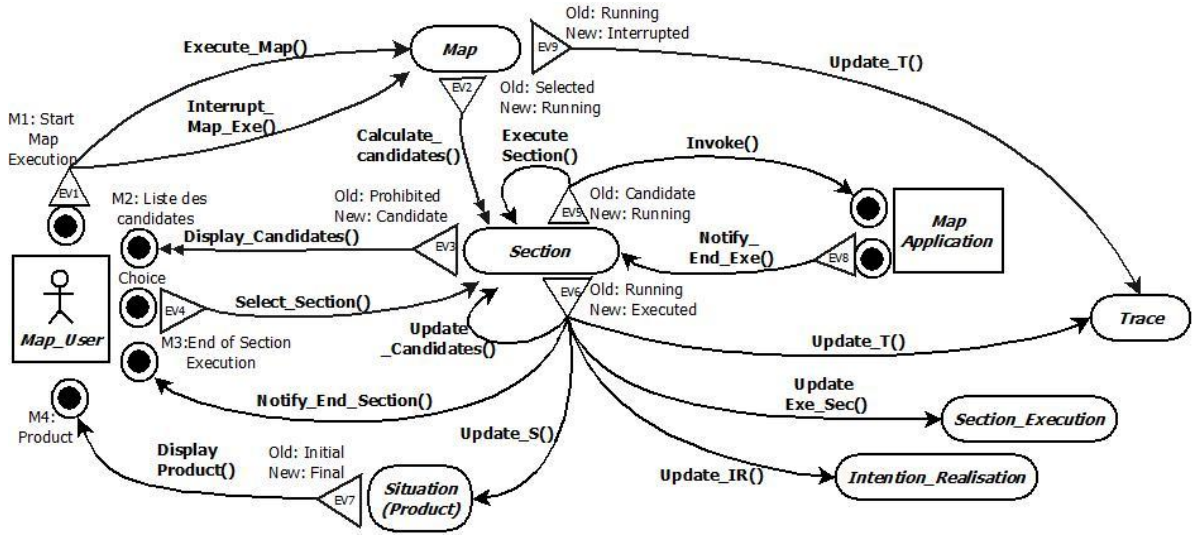

Figure 4. Representation of the dynamic perspective for the Map meta-model

5 DISCUSSION AND CONCLUSION

We have addressed in this position paper the problem of specifying the enactment semantics of a process meta-model. By analyzing the state of the art, we find that the software engineering community has begun addressing this problem. Proposed solutions are based on UML and on the generic MOF meta-model, and they are inspired by the work around the implementation of the SPEM process meta-model. In the method engineering field, approaches are different and the assembly of components is the privileged approach to define new methods. However, we note that the issue of executability is common to both domains of research, in the sense that to be implemented, a methodological component must be specified in detail and its executable semantics have then to be clearly expressed.

By using an adequate modeling notation which combines rigorous behavioral semantics and a clear graphical representation, we showed the contribution of this approach in the expression of process model executability. Our proposal is to be considered as a hybrid approach that combines the advantages of declarative and imperative paradigms for process modeling languages. It allows the construction of dynamic meta-models which have well defined semantics, and which are able to take into account the non-deterministic executable nature of a process model such as the Map. We have to notice here that some of the concepts of the meta-model are considered as object classes in the data perspective, but also as agents in the behavior perspective. This is an important aspect of the behavior perspective since it captures the semantic of the interaction, not only from the information or data point of view but also from the agent point of view. Compared with a meta-programming approach (e.i. Kermeta) or a purely declarative approach (e.i. ConceptBase), our approach highlights graphically the points of interaction between the running process and its environment.

This paper is part of an ongoing research work for the design and the specification of CASE-like software tools to support the Map intentional model. We are convinced that meta-models should deal with concepts of the behavior perspective and not only concepts of data and process perspectives, especially if a model-driven execution tool is to be derived from it. We are currently studying, testing and comparing various meta-modeling environments (Kermeta, MetaEdit, ConceptBase) to assess the relevance of such meta-CASE approaches for building a software support tool. The work presented here suggests that the meta-modeling approaches are multiple and complementary, but suggest also that they are unable to take into account all the requirements of the designer in terms of graphical representation, in terms of expressing correctly the interactive semantics of process models, and in terms of formal and detailed expression of executability.

REFERENCES

Assar, S., Ben Achour, C. & Si-Said, S. 2000. 'A model for the specification of Information Systems analysis process (in French)'. In Proc. *13th INFORSID conference*, Lyon, France.

Bendraou, R., Combemale, B., Cregut, X. & Gervais, M. P. 2007. 'Definition of an Executable SPEM 2.0'. In Proceedings *14th Asia-Pacific Software Engineering Conference (APSEC 2007)*, 4-7 Dec. 2007, Nagoya, Japan.

Bendraou, R., Gervais, M.-P. & Blanc, X. 2005. 'UML4SPM: A UML2.0-Based Metamodel for Software Process Modelling'. *In* Briand, L. & Williams, C. (eds.) *Model Driven Engineering Languages and Systems.* Springer. pp. 17-38.

Breton, E. & Bézivin, J. 2001. 'Towards an understanding of model executability'. In Proceedings *Int. Conf. on Formal Ontology in Information Systems - Volume 2001*, Ogunquit, Maine, .

Brinkkemper, S., Lyytinen, K. & Welke, R. (eds.) 1996. *Method engineering: Principles of method construction and tool support*: Chapman and Hall.

Brinkkemper, S., Saeki, M. & Harmsen, F. 2001. 'A Method Engineering Language for the Description of Systems Development Methods'. *In* Dittrich, K., Geppert, A. & Norrie, M. (eds.) *Advanced Information Systems Engineering (CAiSE).* Springer Berlin / Heidelberg. pp. 473-476.

Harmsen, A. & Saeki, M. 1996. 'Comparison of four method engineering languages'. *In* Brinkkemper, S., Lyytinen, K. & Welke, R. (eds.) *Method engineering: principles of method construction and tool support.* Chapman and Hall.

Henderson-Sellers, B. & Ralyté, J. 2010. 'Situational method engineering: state-of-the-art review'. *Journal of Universal Computer Science*, 16(3), pp. 424-478.

Jarke, M., Jeusfeld, M., Nissen, H., Quix, C. & Staudt, M. 2010. 'Metamodelling with Datalog and Classes: ConceptBase at the Age of 21'. *In* Norrie, M. & Grossniklaus, M. (eds.) *Object Databases.* Springer Berlin / Heidelberg. pp. 95-112.

Jeusfeld, M., Jarke, M. & Mylopoulos, J. 2009. *Metamodeling for method engineering*, Cambridge, MA, The MIT Press.

Kelly, S., Lyytinen, K. & Rossi, M. 1996. 'MetaEdit+ A fully configurable multi-user and multi-tool CASE and CAME environment'. *In* Constantopoulos, P., Mylopoulos, J. & Vassiliou, Y. (eds.) *Advanced Information Systems Engineering (CAiSE).* Springer. pp. 1-21.

Kelly, S. & Tolvanen, J. P. 2008. *Domain-specific modeling: enabling full code generation*, Wiley-IEEE Computer Society Press.

Kermeta. *http://www.kermeta.org* [Online, 2011].

MetaCASE. *http://www.metacase.com/* [Online].

Muller, P.-A., Fleurey, F. & Jézéquel, J.-M. 2005. 'Weaving Executability into Object-Oriented Meta-languages'. *In* Briand, L. & Williams, C. (eds.) *Model Driven Engineering Languages and Systems.* Springer. pp. 264-278.

Mylopoulos, J., Borgida, A., Jarke, M. & Koubarakis, M. 1990. 'Telos: Representing knowledge about information systems'. *ACM Transactions on Information Systems*, 8(4).

Olle, T. W., Hagelstein, J., MacDonald, I. G., Rolland, C., Sol, H. G., Van Assche, F. J. M. & Verrijn-Stuart, A. A. 1991. *Information Systems Methodologies: a framework for understanding*, Addison-Wesley.

OMG. 2008. *Software & Systems Process Engineering Meta-Model Specification, Version 2.0 - OMG document formal/2008-04-01* [Online]. Available at: http://www.omg.org/spec/SPEM/2.0/PDF.

Rolland, C. 2007a. 'Capturing System Intentionality with Maps'. *In* Krogstie, J., Opdahl, A. L. & Brinkkemper, S. (eds.) *Conceptual Modelling in Information Systems Engineering.* Springer Berlin Heidelberg. pp. 141-158.

Rolland, C. 2007b. 'Method Engineering: Trends and Challenges (Invited talk)'. *In* Ralyté, J., Brinkkemper, S. & Henderson-Sellers, B. (eds.) *Situational Method Engineering: Fundamentals and Experiences.* Springer Boston. pp. 6-6.

Seidita, V., Ralyté, J., Henderson-Sellers, B., Cossentino, M. & Arni-Bloch, N. 2007. 'A comparison of deontic matrices, maps and activity diagrams for the construction of situational methods'. In Proceedings *CAiSE Forum, 19th Int. Conf. on Advanced Information Systems Engineering*, 11-15 June 2007, Trondheim, Norway.

Sprinkle, J., Rumpe, B., Vangheluwe, H. & Karsai, G. 2011. 'Metamodelling: State of the Art and Research Challenges'. *In* Giese, H., Karsai, G., Lee, E., Rumpe, B. & Schätz, B. (eds.) *Model-Based Engineering of Embedded Real-Time Systems.* Springer Berlin / Heidelberg. pp. 57-76.