SINTEF A231 – Unrestricted

REPORT

Metamodels for Model-Driven Development - Requirement and Usage

Ida Solheim, Svein G. Johnsen, Tor Neple

SINTEF Information and Communication Systems

August 2006

SINTEF

SINTEF ICT

Address: NO-7465 Trondheim, NORWAY
Location: Forskningsveien 1
Telephone: +47 22 06 73 00
Fax: +47 22 06 73 50

Enterprise No.: NO 948 007 029 MVA

SINTEF REPORT

TITLE

Metamodels for Model-Driven Development - Requirement and Usage

AUTHOR(S)

Ida Solheim, Svein G. Johnsen, Tor Neple

CLIENT(S)

The Modelware project (FP6-2003-IST-2-511731)

REPORT NO.	CLASSIFICATION	CLIENTS REF.	
SINTEF A231	Open	Wp2 – Process and Methodologies	
CLASS. THIS PAGE	ISBN	PROJECT NO.	NO. OF PAGES/APPENDICES
Open	82-14-04035-3	90B20920	17 / 0
ELECTRONIC FILE CODE		PROJECT MANAGER (NAME, SIGN.)	CHECKED BY (NAME, SIGN.)
Metamodels for MDD (SINTEF report).doc		Jan Øyvind Aagedal	Jan Øyvind Aagedal
FILE CODE	DATE	APPROVED BY (NAME, POSITION, SIGN.)	
	2006-08-11	Bjørn Skjellaug, Research director	

ABSTRACT

Within software engineering, there are several standardisation activities aiming at developing the best general process metamodel. Two candidate standards are SPEM (of OMG) and SMSDM (of ISO/IEC). This paper studies each of these metamodels with respect to model-driven development (MDD), evaluating their domain appropriateness and comprehensibility appropriateness. The conclusion is that both metamodels are appropriate for the MDD domain, but SPEM is less comprehensible than SMSDM. A third criterion, technical actor appropriateness, should be evaluated for these metamodels when used in automated process modeling tools. Such tools should enable the combination of different process frameworks, by instantiating reusable process elements and tying them together to form tailored process models. Metamodel interoperability is required for this purpose.

KEYWORDS	ENGLISH	NORWEGIAN
GROUP 1	ICT	IKT
GROUP 2	Information Systems	Informasjonssystemer
SELECTED BY AUTHOR	Model-Driven Development	Modelldrevet utvikling

TABLE OF CONTENTS

1 Introduction

1.1 Background

Process modelling has been around for some years, helping organizations and employees to identify, analyze and improve their processes. A model is based on a (more or less explicit) metamodel, which prescribes constructs and rules for creating a model. This paper focuses on metamodels for process modelling. It is not concerned about the modelling of general processes, but of processes required to perform model-driven software development (MDD). In MDD, models are the prime artefacts. That means, models are in use throughout the whole production chain – from the early capture of user requirements to the production of executable code. Model reuse is essential, and also model transformations, which preferably should be automated. Indeed, tool support is by many considered a prerequisite for successful MDD (e.g. [2], [3]).

Although MDD has been practiced for years (e.g. [4]), it did not gain ground until the Object Management Group (OMG) launched its Model-Driven Architecture ® (MDA ®) initiative [5]. Being "an approach to using models in software development" [5], MDA has boosted the development of tools and thereby (semi)automation of program development and maintenance. MDA motivates MDD processes with the following characteristics:

- Many activities have models as input, or output, or both.
- Several of these activities are model transformations (while others are model analysis, model verification etc.).
- A transformation takes a model as input and produces a model, or text, as output. During the transformation, the output model is supplied with domain-related information not present in the input model. An example of such a domain is the *platform* concept, often used for "implementation platform".

Appropriate process models may be useful for software developers aiming at adopting MDD. General process metamodels exist; for the time being at least two metamodels for software engineering are being standardized by their respective standardization bodies:

- *Software Process Engineering Metamodel* (SPEM) [6], an industry standard being developed by the Object Management Group (OMG). The scope of SPEM is to define a "minimal set of process modelling elements necessary to describe any software development process".
- *Standard metamodel for software development methodologies* (SMSDM) [7]. This Australian standard has been accepted as a work item of ISO/IEC JTC1 [8], in which it, at the time of writing, has the status of a working draft [9]. The scope of SMSDM is to establish a "formal framework for the definition and extension of software development methodologies".

Both SPEM and SMSDM claim to be appropriate for all kinds of software engineering processes. SPEM is even supported by some off-the-shelf tools; among which are IRIS Process Author [10], Rational Rose XDE Developer Plus [11], and Objecteering/UML Enterprise Edition [12].

1.2 Questions

An MDD process is assumed to be a special case of a general software development process. Therefore, a metamodel for MDD processes would definitely resemble a metamodel for any other software development process. However, there may exist specific requirements to MDD process

modelling, which are not catered for by general-purpose process metamodels. Hence, the authors pose the following questions concerning a metamodel for MDD processes:

1. Which requirements must a metamodel fulfil in order to support the modelling of MDD processes?
2. Which metamodel is more appropriate for modelling MDD processes, SPEM or SMSDM?
3. Which role does an MDD metamodel play with respect to a process framework?

1.3 About this Paper

This work is part of the EU-funded R&D project MODELWARE [13], in which it represents a starting point for the elaboration of an MDD process framework. This paper identifies a few important (but not exhaustive) requirements to an MDD process metamodel (chapter 2). Then, referring to an example process (chapter 3), SPEM and SMSDM are investigated with respect to the selected requirements (chapters 4 and 5). Further, the usage of an MDD process metamodel is discussed, relating the metamodel to a process framework (chapter 6). The paper concludes with the results of the evaluation (chapter 7) and suggests further work with respect to metamodel investigation, and to development of an MDD process framework.

2 Requirements and Approach

A metamodel suitable for modelling MDD processes need to fulfil the following requirements:

a. It must be possible to specify a *transformation* as an artefact, and a *model* as an artefact, and relate those by input or output relations (or both). Model transformations are central in MDD, and ongoing work tries to cope with their various complexities (e.g. Bezivin et al. [3]).
b. It must be possible to include *traceability* information as output from a transformation. Such information will ensure that model elements are traceable through all transformations, backwards and forwards.
c. It must be possible to distinguish between manual and automatic (e.g. tool-supported) activities.
d. The metamodel must have a graphical part that eases its *understandability* by humans.

Based on an example, the following chapters evaluate the two standard process metamodels against the above requirements. The evaluation is based on the framework of Krogstie and Sølvberg [14]. This framework aims at evaluating the qualities of models and modelling languages, e.g. the UML. According to this framework, a modelling language may be evaluated according to several types of appropriateness, e.g. domain appropriateness, organizational appropriateness, comprehensibility appropriateness, etc. (Fig. 1). The UML has already been evaluated according to this framework [1].

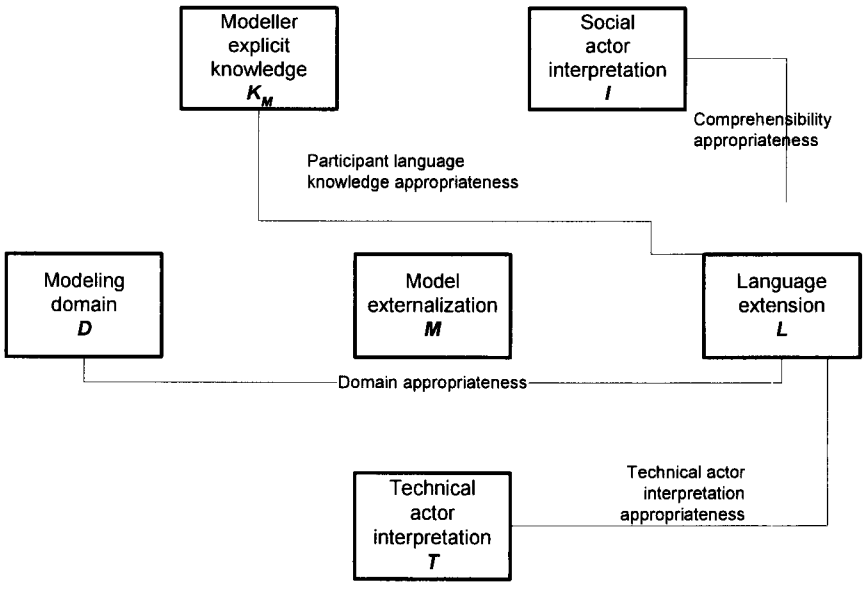

Fig. 1 Kinds of language appropriateness. Rendered from [1] by courtesy of the author.

The appropriateness of SPEM and SMSDM to support general software processes will not be discussed here, but are left for their respective standardization environments. What to be investigated in the following are their abilities to support MDD. Therefore, only these kinds of appropriateness will be investigated:

- *Domain appropriateness* with respect to the MDD domain (requirements a—c)
- *Comprehensibility appropriateness* with respect to modelling MDD-specific processes (requirement e). This appropriateness is relevant to vendors of process modelling tools. In cases where no such tool is available, the process/method engineer [15] will have to use the metamodel directly and hence need to understand it.

Technical actor appropriateness is relevant to evaluation after the metamodel has been realized in a process modelling tool. Such evaluation is left for further work.

3 An Example Process

This chapter presents a selected example of an MDD process that will be used to discuss the appropriateness of SPEM and SMSDM metamodels.

In model-driven development, one characteristic process is the *transformation* from a platform-independent model (PIM) to a platform-specific model (PSM). (What distinguishes a PIM from a PSM is described inter alia in [5].) Hence, one needs a kind of *activity* or *work definition* that takes a *model* as input and produces another model as output. The selected transformation is assumed to know the PIM's metamodel and the PSM's metamodel. A *transformation record* is supposed to be tied to each iteration of this process and stored in some repository for later reference. It is assumed that *traceability* information exists as part of this record, or in addition to it. Such traces provide links from input model elements to output model elements and thus help verify that the transformation has performed as expected.

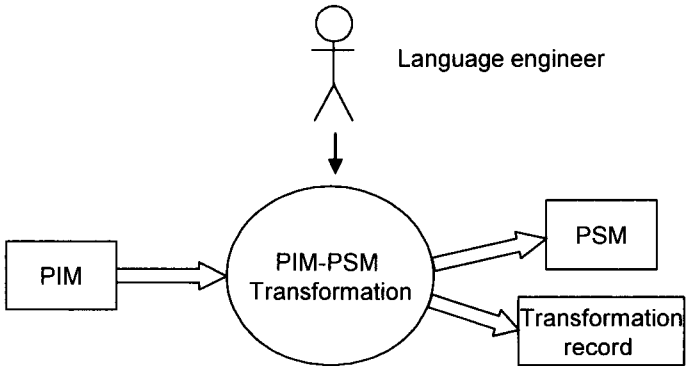

Fig. 2 An informal sketch of a PIM-to-PSM transformation.

Fig. 2 depicts an informal sketch of the example. In the next chapters, this example process will be modelled by means of SPEM and SMSDM.

4 Evaluating SPEM

This work uses SPEM 1.1, which has undergone several improvements compared to its predecessor. SPEM is defined as an extended subset of UML's metamodel, and is also defined as a UML profile. The fact that SPEM is part of UML, prescribes the usage of UML´s metamodel wherever appropriate. Therefore, some of the model elements constituting SPEM inherit from model elements already defined in UML´s metamodel. References are made to the UML packages "Data_Types", "Core", "Actions", "State_Machines", "Activity_Graphs" and "Model_Management".

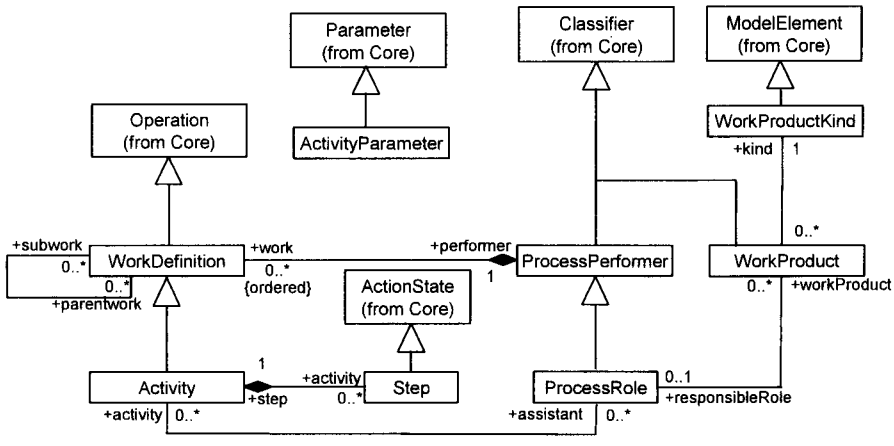

Fig. 3 A relevant part of SPEM's class diagram.

The SPEM metamodel is expressed in a class diagram spanning several figures in the specification. The part rendered in Fig. 3 is relevant to this study. Inheritances to UML's metamodel are also shown.

Fig. 3 shows some important relationships between ProcessPerformers, Activites and WorkProducts, but does not show all of them. For example, based on Fig. 3 alone, one cannot deduce a direct connection between an activity and the work products resulting from that activity. Further, Fig. 3 does not show which artefacts may be input to, or output from, an activity. Nor is there an explicit sequence between activities (or steps). However, by carefully reading the text accompanying Fig. 3, one may start nesting inheritance relationships from SPEM to UML's metamodel. This way, we may verify that SPEM 1.1:

a. Allows the use of models as artefacts and as activity input/output. This is done by means of the WorkProduct class. The Activity class is associated with WorkProducts via the ActivityParameter class. This class tells which WorkProducts are inputs to, and which are outputs from, the Activity. Model transformations are defined by means of the Activity class.
b. Enables specifications of a transformation record. This is done by using the WorkProduct class.
c. Allows distinction between manual and automatic activities. This may be obtained by using the class Guidance, which is associated with the Activity class.

Hence, the requirements a—c are satisfied, and we may conclude that SPEM 1.1 has sufficient domain appropriateness for the MDD domain. The example process may be modelled as follows using SPEM's UML profile:

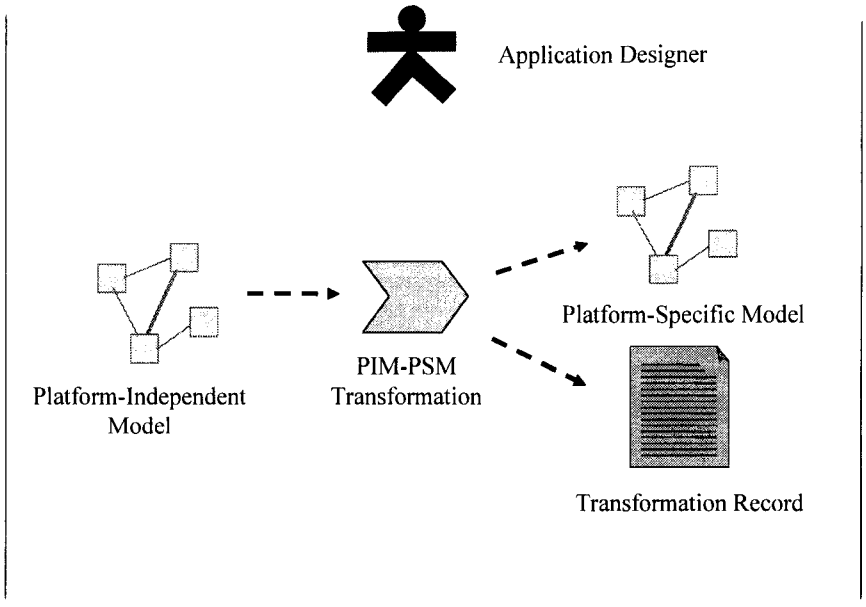

Fig. 4 PIM-to-PSM transformation expressed in a swim line diagram using SPEM's UML profile.

When it comes to requirement d, comprehensibility appropriateness, our impressions are varied. On the one hand, SPEM's strong tie to UML's metamodel may be regarded as a strength. Since SPEM is an extended subset of UML's metamodel, it reuses model constructs already present in the UML. This way, existing model constructs – which presumably have undergone thorough consideration – need not be reinvented or redefined.

On the other hand, the extensive dependence on UML's metamodel makes SPEM particularly complex and consequently difficult to comprehend. For this reason, we may conclude that the SPEM 1.1 is weak with regard to comprehensibility appropriateness.

5 Evaluating SMSDM

Already a de jure Australian standard, SMSDM has been adopted as a working draft for standardization by ISO/IEC JTC1 [9]. A main difference from SPEM is that SMSDM does not include parts of UML's metamodel. However, SMSDM *uses* the UML to express the process metamodel. Similarly to SPEM, this is done in the form of a UML class diagram, which is too large to fit into one figure. The most relevant part is rendered in Fig. 5[1].

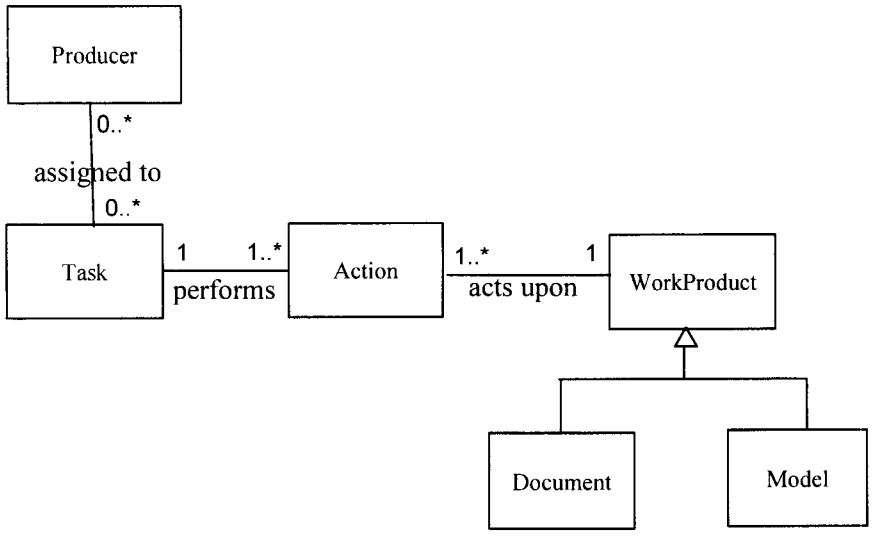

Fig. 5 A relevant part of SMSDM's class diagram

After studying the SMSDM class models and the accompanying text, we are able to verify that SMSDM:

a. Allows the use of models as artefacts and as activity input/output. This is done by using the Model class, which is a subclass of the more general WorkProduct class. The Model transformations are defined by means of the Task class. The Action class holds information about how the WorkProducts are handled in Model transformations.
b. Enables the specification of transformation record. This is done by using the Document class (alternatively, the Model class), which is a subclass of the WorkProduct class.
c. Allows distinction between manual and automatic activities. This may be obtained by using the class Technique, which is associated with the Task class.

Hence, the requirements a—c are satisfied, and we may conclude that SMSDM 1.1 has sufficient domain appropriateness for MDD domain. The example process may be modelled in SMSDM as follows:

[1] The figure does not contain the "clabject" (class + object) concept, even though the authors appreciate it as a very convenient metamodel construct. It is omitted in order to concentrate our discussion on SMSDM's appropriateness applied to the modeling of MDD processes.

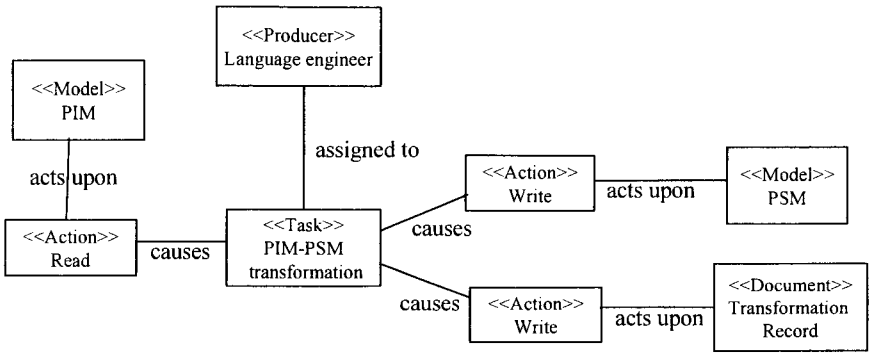

Fig. 6 PIM-to-PSM transformation expressed in a class diagram based on SMSDM.

SMSDM appears understandable and straightforward. Unlike SPEM, it is self-contained in that it does not reuse definitions from UML's metamodel. This is an important reason for its comprehensibility. We conclude that SMSDM fulfils requirement d, comprehensibility appropriateness.

6 Usage of a Process Metamodel in Model-Driven Development

6.1 A Metamodel's Role in a Process Framework

A process framework (PF) is a repository of reusable process model elements (process patterns, templates, etc...) to be used for building process models. Given that the elements in a PF are expressed in a well-defined formal language, the PF must be based on a metamodel defining this language. Several PFs might be in use in concert, each offering process model elements representing specific aspects or domains, e.g. organization-specific aspects, product-specific aspects etc.

6.2 A Process Framework for MDD

One domain of specific interest to be covered by a PF is the MDD domain. An MDD PF will contain only MDD-specific elements, which are process model elements to be used to build software engineering process models in compliance with MDD principles.

Software engineering companies may have their own standard development processes. When starting to use MDD, a company may not want to redefine its PF, but rather plug into it appropriate MDD-specific processes from an MDD PF. This means to select the appropriate elements from the two frameworks, prepare these elements (e.g. instantiate a process pattern) and compose them to form a process model by the means of a modelling tool. This model will then reflect the various aspects of the different PFs in use.

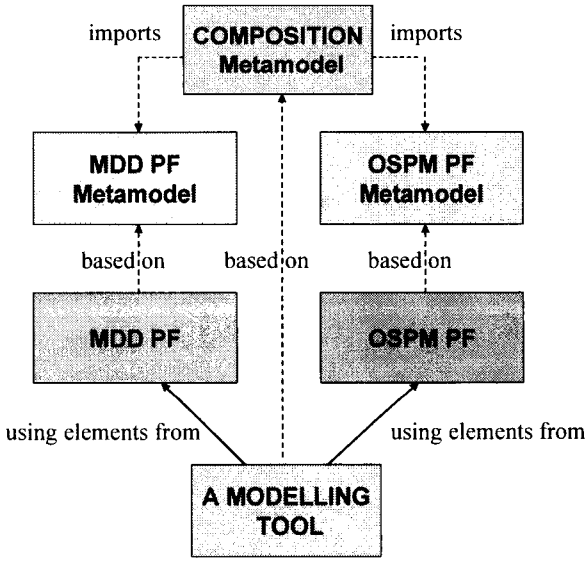

Fig. 7 A process model may be composed of elements from two different PFs – if their respective metamodels are interoperable

Fig. 7 sketches the use of an MDD PF together with an OSPM (Organisational Standard Process Model). Both PFs are supposed to be based on metamodels. Elements of the PFs are composed into a complete software engineering process model by means of an appropriate modelling tool. The resulting process model should reflect both the MDD aspects and the aspects of the OSPM.

For PFs to be combined in this way, the elements from the one PF must be "plug-compatible" with elements from the other PF. To achieve such compatibility, harmonization is required on two levels:

- Metamodel level: The metamodels used by the PFs must share a sufficient set of constructs and rules.
- Model level: The elements of the different PFs must have a common ontology and common semantic rules.

The harmonization on the metamodel level may be achieved by creating a metamodel (called the *composition* metamodel in Fig. 7) with the purpose of bridging the MDD PF metamodel with the OSPM metamodel. On the model level, the modelling tool has to base its language on the "composition metamodel". This way, the tool may enable composition of a new process model based on model elements from both PFs.

6.3 Related Work

There is an abundance of work on software process modelling. However, only a few selected papers are referenced here, in order to put the work of this paper into context. The traditional gap between structural/object-oriented metamodeling and process metamodeling has been addressed by several authors. Dori and Rheinhartz-Berger suggest to bridge this gap by introducing the Object-Process Methodology containing an OPM-based metamodel. Henderson-Sellers et al. give a similar contribution in their introduction to SMSDM [7]. Gnatz et al. [16] suggest a metamodel by which to describe both work artefacts, process artefacts, and relations between them. This metamodel is supposed to be the basis for software development processes in which initial artefact descriptions are tailored and reused during iterative and evolutionary process improvement. Caivano and Visaggio [17] emphasize reusable process descriptions, suggesting a framework based on process patterns, but no explicit metamodel. Breton and Bezivin [18] address coupling between metamodels.

7 Conclusion and Further Work

This study has evaluated the two standardized metamodels SPEM and SMSDM with respect to usage within the domain of model-driven software development. Two evaluation criteria have been used: domain appropriateness and comprehensibility appropriateness. The following conclusions were drawn:

	Domain appropriateness	***Comprehensibility appropriateness***
SPEM 1.1	The SPEM metamodel's ability to express MDD-specifics has been verified through a worked example.	SPEM has a low score on comprehensibility due to the complexity of inheritance relations to UML's metamodel.
SMSDM	SMSDM's ability to express MDD-specifics has been verified through a worked example.	SMSDM has a high score on comprehensibility due to its straightforward and easy-to-understand class diagrams and textual explanations.

A metamodel's successful usage is to a large degree dependent on automatic tools that support it. An operational MDD process framework would provide a tool to guide software developers through MDD processes. The metamodel of this MDD framework may be SPEM 1.1 or SMSDM. When implemented in a tool, the metamodel should be evaluated with respect to a criterion not studied here, namely *technical actor appropriateness* (Fig. 1). This appropriateness is a means to achieve syntactic, semantic and pragmatic quality of the models created by the tool [1]. Such evaluation is left for further work.

More important, there is a need to further elaborate the idea of an MDD process framework; its contents, appropriateness, usage and – not least – tool support. This is another work item of the MODELWARE project.

References

1. Krogstie, J.: Evaluating UML Using a Generic Quality Framework. In: Favre, L. (ed.): UML and the Unified Process. IRM Press (2003) 1-22.
2. Alanen, M., et al.: Model Driven Engineering: A Position Paper. 1st International Workshop on Model-Based Methodologies for Pervasive and Embedded Software, MOMPES'04 (2004)
3. Bézivin, J., et al.: The ATL Tranformation-based Model Management Framework. Research report. IRIN, Université de Nantes, Nantes, France (2003)
4. Grønmo, R., et al.: DISGIS: An Interoperability Framework for GIS - Using the ISO/TC 211 Model-based Approach. Global Spatial Data Infrastructure (GSDI) 4. Cape Town, South Africa (2000)
5. Object Management Group: MDA Guide. Ver. 1.0.1. http://www.omg.org/docs/omg/03-06-01.pdf (2003).
6. Object Management Group: Software Process Engineering Metamodel Specification. Ver. 1.1. (2005).
7. Standards Australia: Standard metamodel for software development methodologies. AS 4651-2004 (2004)
8. ISO/IEC JTC 1: SC 7 Proposed New Work Proposal for Standard Metamodel for Software Development Methodologies. Ver. N7466. (2004).
9. ISO/IEC: Standard Metamodel for Development Methodologies in Information-Based Domains. Ver. 1.3. Working draft 24745 (2005).
10. Oscellus, Inc.: IRIS Process Author. http://www.osellus.com/products/irispa.html
11. IBM: Rational Rose XDE Developer Plus. http://www-306.ibm.com/software/awdtools/developer/plus/
12. SOFTEAM: Objecteering/UML Enterprise Edition. http://www.objecteering.com/packaging_enterprise_edition.php
13. IST Project 511731: MODELWARE. Modeling solution for software systems. http://www.modelware-ist.org/ (2004-2006)
14. Krogstie, J. and Sølvberg, A.: Information systems engineering - Conceptual modeling in a quality perspective. Kompendiumforlaget. Trondheim, Norway (2003)
15. Aagedal, J.Ø. and Solheim, I.: New Roles in Model-Driven Development. Second European Workshop on Model Driven Architecture (MDA), EWMDA-2. Canterbury, UK (2004)
16. Gnatz, M., et al.: The Living Software Development Process. In: Software Quality Professional, Vol. 5, No. 3 (2003) 4-16
17. Caivano, D. and Visaggio, C.A.: Process Diversity and how Practitioners Can Manage It. In: UPGRADE. The European Journal for the Informatics Professional, Vol. V, No. 5 (2004) 59-66
18. Breton, E. and Bézivin, J.: Model-Driven Process Engineering. Annual International Computer Software and Applications Conference, COMPSAC. Chicago (2001)