Applying The Basic Principles of Model Engineering to The Field of Process Engineering

Jean Bézivin and Erwan Breton

A new information system landscape is emerging that will probably be more model-centred than object-oriented, characterized by many models of low granularity and high abstraction. These models may describe various aspects of a system such as software product properties, static and dynamic business organization, non-functional requirements, middleware platforms, software processes, and many more. Each model represents some particular point of view of a given system, existing or in construction, and this model conforms to a precise metamodel or DSL (Domain Specific Language). In this paper we present some advantages of using the unification framework of model engineering to deal with the various facets of process engineering. As the view of the software life-cycle is progressively shifting from a simple definition and composition of objects to a sequence of model transformations, the need to characterize this by a precise process is becoming urgent. Description of software artifacts, processes and transformations may all be uniformly captured by different forms of models. This approach provides a regular framework where business and software production process models are going to play an increasingly important role. In this paper we illustrate some possibilities of model-based process engineering.

Keywords: Domain Specific Languages, MDA, Model-Based Process Engineering, MS-Project, SPEM.

1 Introduction

As stated in the seminal work of Osterweil [10], *"Software processes are software too"*. Therefore, the same technologies may be used for supporting both software artifacts and software processes. Among these technologies, Model Driven Engineering (MDE) has recently taken an important place. One of the best known families of MDE is the Model Driven Architecture (MDA™ organization) proposed by OMG (Object Management Group) in November 2000 [11]. As a consequence we can today envisage a new situation where software products are models and software processes are models too. At a lower level of abstraction, object technology tried twenty years ago to unify the field of software engineering by considering objects as first-class entities. The new trend of model engineering considers instead models as first-class entities. A model is a representation of a given system and is written in the language of its metamodel (it conforms to its metamodel). These relations of *representation* and *conformance* are characteristics of new model engineering techniques [2].

MDA gives a central role to models and metamodels. Taking account of the various aspects of information system engineering can only be achieved through the well-organized management of a complex lattice of related metamodels. The emerging tendency is to separate business knowledge expression from implementation as two separate aspects captured by specific models. The invariants of a business domain may be held apart from the execution code, preserving them from technology obsolescence. The information of a complex enterprise system originates from various sources (system engineers, managers, software engineers, quality engineers, etc). All the corresponding domains may be expressed using their own specific languages.

This paper discusses process engineering using MDA-based techniques. We present in Section 2 a brief overview of MDA principles. In Section 3, we focus on software process-related standards. Building on this, we show the practical interest and

Jean Bézivin is Professor of Computer Science at the University of Nantes, France, member of the newly created ATLAS research group in Nantes (INRIA & CNRS/LINA). He has been very active in Europe in the Object-Oriented community, starting the ECOOP series of conference (with P. Cointe), the TOOLS series of conferences (with B. Meyer), and more recently the <<UML>> series of conferences (with P.-A. Muller). His present research interests include legacy reverse engineering, general model engineering (MDE/MDA™) and more especially model-transformation languages and frameworks. He is currently involved in the Modelware and Interop european projects. <Jean.Bezivin@lina.univ-nantes.fr>

Erwan Breton is a Consultant in the Sodifrance/SoftMaint company in Nantes, France, working on the subjects of process modelling and workflow. He is presently in charge of the R&D strategy for the company. After obtaining a Master degree from the University of Lille 1, France, he got a PhD in Computer Science from the University of Nantes. He has published several papers in the area of model-based process engineering. <ebreton@sodifrance.fr>

the industrial applicability of this vision in Section 4, based on a simple example. General considerations on present capabilities and limits of the approach are summarized in the conclusion together with some current perspectives on model-driven platforms.

2 Model Driven Engineering and The OMG/MDA Organization

In the IBM manifesto [4], the principles of model engineering are presented as the three vertices of a triangle:

- *Direct representation*, meaning that for each aspect of a system under construction or maintenance, there is a domain specific language (DSL) dedicated to handling this aspect;
- *Automation*, meaning for example that a mapping from DSL programs to executable frameworks may be automatically handled, mainly by model transformation procedures;
- *Open standards*, on which these DSLs and transformations will be based to allow the cooperation of various tools for capturing and operating on models.

Each DSL corresponds to a given metamodel. In the OMG MDA stack, all metamodels conform to a unique meta-meta-model named the MOF (Meta-Object Facility). The MOF is a language to write metamodels. Furthermore the MOF has a number of common facilities to deal with metamodels and their models. Some of them are related to establishing bridges to other technical spaces. MDA is one such technical space that could be called modelware but middleware, grammarware, executable programming languages, XML (eXtensible Markup Language) document management, data bases, Semantic Web and ontology engineering are other examples of technical spaces. There is a standard mapping between MDA and CORBA/IDL middleware (CMI, Corba Model Interchange [7]) but there are also other mappings to XML document management (XMI, XML Model Interchange) and to the Java technical space (JMI, Java Model Interchange).

The so-called OMG MDA stack is thus composed of the three following layers:

- At level M3, the MOF defines a representation system based on constrained graphs. These graphs are similar to UML (Unified Modelling Language) class diagrams. In addition to several facilities mentioned above, an assertion/navigation language named OCL is also provided at this level. The main idea is that level M3 contains all that is domain independent.
- Level M2 corresponds to the domain-dependent definitions. It is composed of a set of DSLs, each defined by its metamodel. This collection of metamodels is rapidly growing and may serve many purposes. Among the most known elements at this level, we may list the UML, SPEM (Software Process Engineering Metamodel), CWM (Common Warehouse metamodel), EDOC (Enterprise Distributed Object Computing metamodel).
- At level M1 various extensional definitions corresponding to intentional definitions of level M2 may be found. This is the level of models, each model conforming to a given M2 level metamodel.

The issue of tooling is important in this MDA engineering evolution. Developing an industrial tool is costly and takes time. Since there was no available tool and a very narrow market at level M3, the trick has been to consider temporarily an alternate way for a MOF editing tool. Instead of defining DSLs as full fledged MOF metamodels, it was made possible to define them as 'specializations' of UML, as so-called profiles. Two parallel competing techniques could then be used for defining the DSLs from scratch or as extensions of UML. The industrial success of UML allowed many industrial or open-source tools to become widely available in a short time, giving an easy path to defining DSLs as UML profiles. Some standards like the SPEM were even jointly defined as a full-fledged MOF metamodel and as UML profile at the same time. Many bridges were also provided to convert between UML profiles and MOF metamodels. Today both techniques are still used, the direct MOF metamodelling way being more widely used to define small well focused and precisely defined DSLs.

Looking backwards, the main turn was taken at OMG when the Analysis and Design Task Force (ADTF) decided to give up the quest for a Unified Method (i.e. a unique object-oriented software development method). This goal was considered as too ambitious and instead the OMG concentrated on defining a DSL for describing object-oriented software artifacts. This DSL was rapidly named UML. Its acceptance was followed later by the definition of another DSL for describing related software processes. Initially known as UPM (for Unified Process Model) it was later renamed as SPEM. Having two related languages, one (UML) for defining the basic software artifacts and one (SPEM) for defining the process for using or producing these artifacts could be considered, as an afterthought, as one of the major achievements of the ADTF definition group. Not all problems of relations between these two independent but related metamodels are yet solved, nor even clearly understood, but the original idea of separating the DSL for software products and the DSL for software processes was clearly an important step that later lead to the foundation of the MDA proposal. This is still a central motivating example and source of inspiration for studying aspect separation and weaving in the context of MDE.

The MDA is often presented as the separation from the platform-independent and platform dependent aspects of software systems. As a matter of fact this corresponds to the definition of DSLs for enterprise description (e.g. EDOC) and DSLs for platform descriptions (e.g. CCM: CORBA Component Models). But programs in these DSLs are intended to be interwoven in order to produce so-called PSMs (Platform Specific Models) from PIMs (Platform Independent Models). The real scope of MDA is indeed larger and encompasses aspect separation issues other than only business and platform aspects. One of them is for example the product/process aspect separation and weaving, either at the level of business (i.e. weaving models of business objects, rules and processes) and at the level of software production (i.e. UML and SPEM as mentioned earlier).

The generation of PSMs from PIMs should be made as automatic and generic as possible. This means that the target platform could be easily changed. This also means that we need

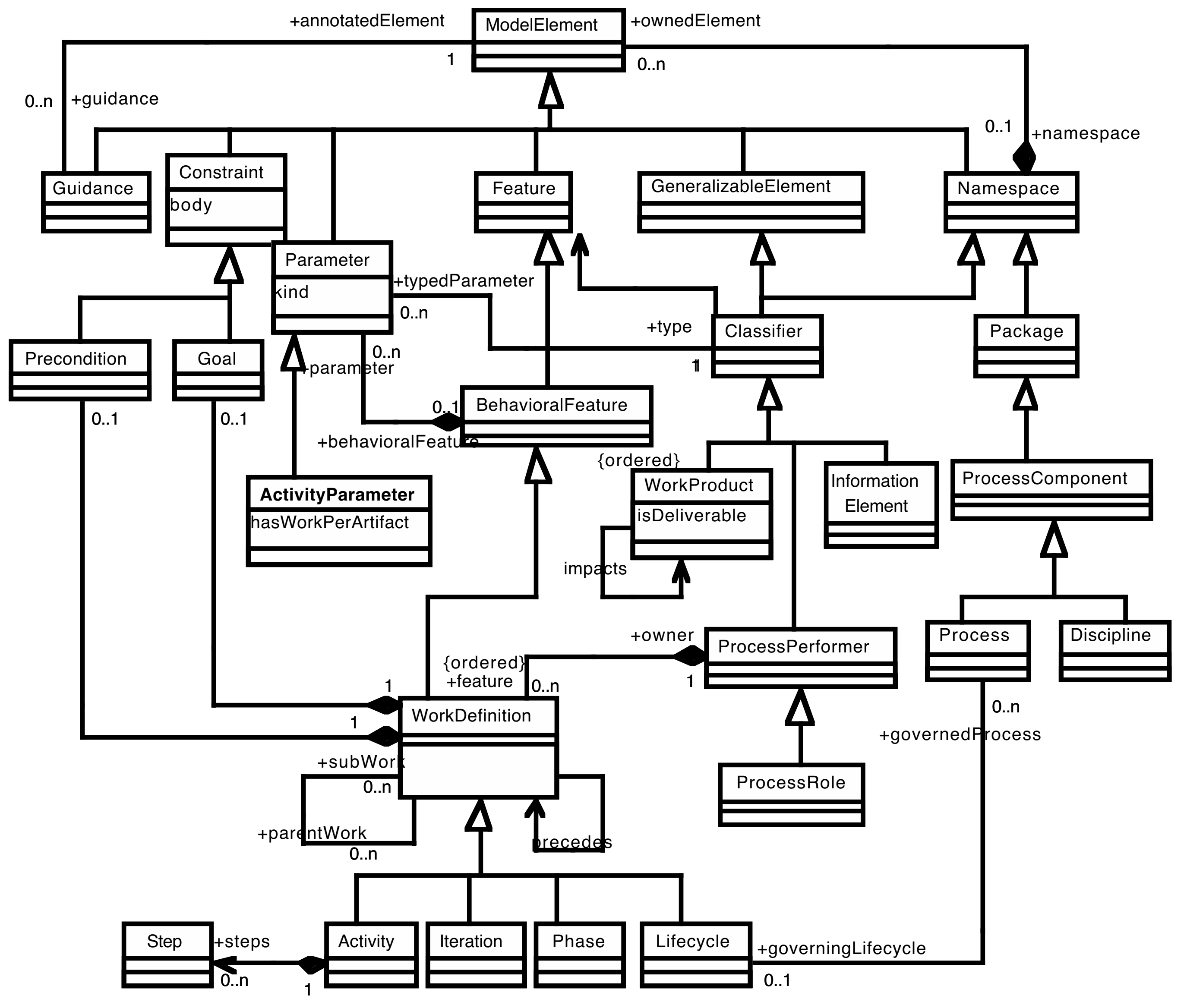

Figure 1: Fragment of SPEM Meta-model.

some transformation language – yet another DSL – to translate between any pairs of source and target DSLs. In order to provide this generic technology, the OMG has launched the MOF/QVT request for proposal (Queries/View/Transformation). The MIA [9] and the ATL [1] languages and systems may be viewed as QVT-compliant transformation systems.

3 Process Modelling

There are several specifications of the OMG that address the issue of process definition. UML introduces the concept of the activity graph. EDOC defines a process as a component with an internal choreography. SPEM is more specifically dedicated to software process description. Other recommendations deal with various forms of business processes.

SPEM defines a generic software process DSL (see Figure 1). A process is described as a set of work items (Work Definition). These work items are performed by process roles. The results of work definition are work products. A work product may be typed by another product DSL or part of, like UML or UML use cases or collaboration diagrams. For the time being the explicit references between SPEM models and software product models are sometimes left implicit.

Beyond these various OMG DSLs, other technical spaces also have their own DSLs in the field of process management, the most active being the XML document space with such proposals as ebXML (Electronic Business XML Initiative) or BPEL4WS (Business Process Execution Language for Web Services) for example. One example is BPMN (Business Process Modelling Notation) [12], but we may also mention IDEF, Activity-Decision Flow (ADF) diagrams, RosettaNet, LOVeM, Event-Process Chains (EPCs), ABC (Activity-Based Costing), REA (Resource/Event/Action), etc.

Studying all these notations for process definitions, we came to several conclusions in [5], and in particular:

- that a common description of all these notations with a unique framework, based for example on the MOF, could be useful to compare them. We expressed several of these formalisms as explicit MOF metamodels;
- that this homogeneous description of different process DSLs with the same metamodelling language could suggest using a common kernel (similar to the PIF, Process Interchange Format, or to the PSL, Process Specification Language) on top of which several extensions could be built;
- that similar concepts were often expressed in different ways in various technical spaces;
- that a clear expression with a precise metamodelling notation like the MOF could lead to well controlled modularity and extensibility. To give an example of this, the SPEM intended to define forward-based software process could be extended to take into account backward-based process dealing for example with legacy recovery and software modernization. The MDA approach is currently applied to both approaches, taking into account not only the platforms of the present and the platforms of the future but also the platforms of the past (COBOL, RPG, PL/1, etc.). Transforming a legacy PSM into a PIM is however probably harder than transforming a PIM into a PSM for EJBs, DotNet or the Grid;
- that process management encompasses different needs (process definition, project planning and enactment) that may be addressed with different tools (modeler, planning tools, workflow systems), which may be integrated using model transformation.

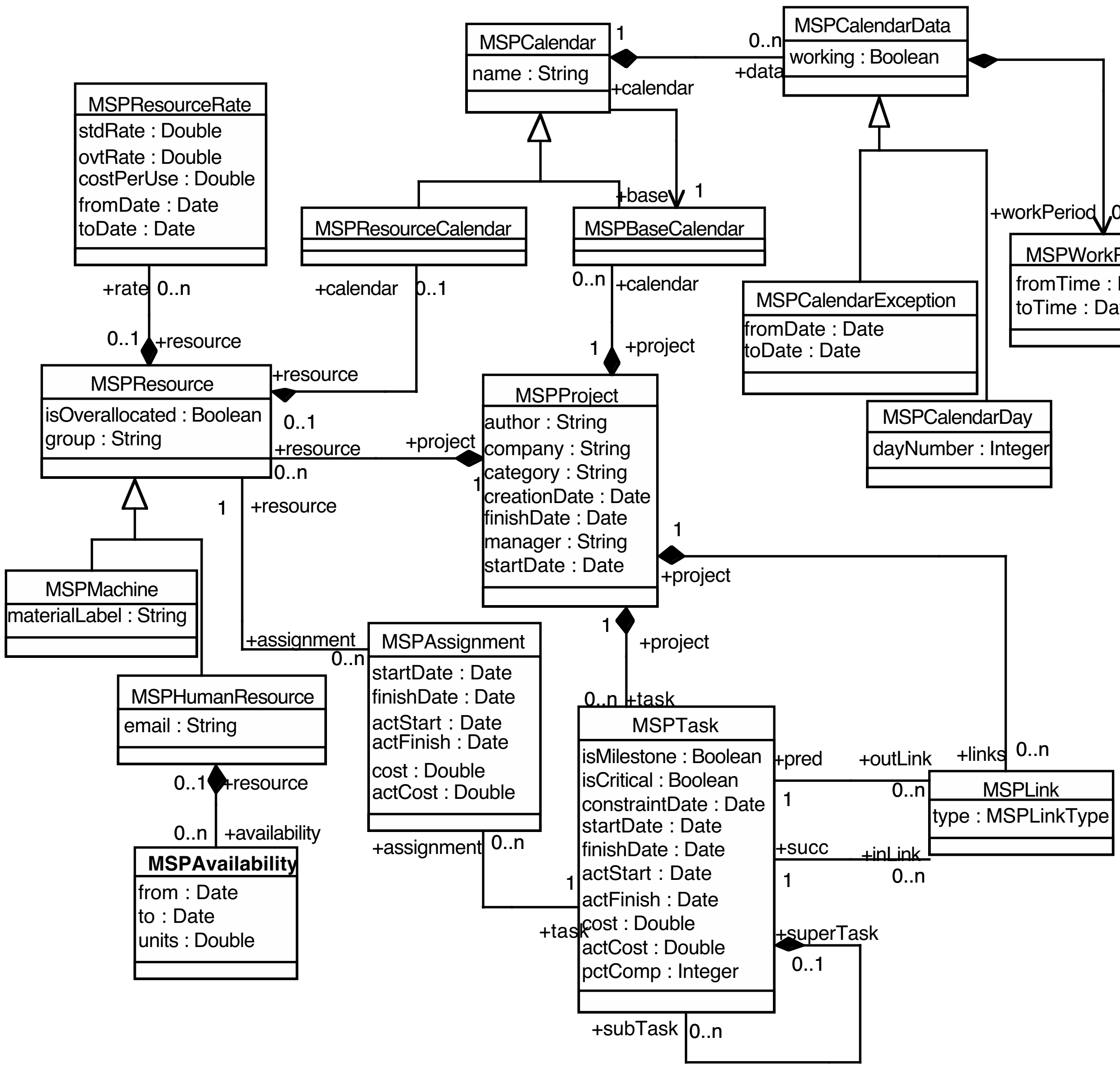

Figure 2: Fragment of MS-Project Meta-model.

It became also obvious that the expression of processes was often a matter of implementation platforms. Mapping abstract processes, expressed in DSLs based on a common metamodelling language like the MOF, onto these platforms could be of great interest. In other words we applied the MDA principle of PIM and PSM to the process themselves. The example proposed in the next section is an illustration of this possibility. Many other examples could have been provided like translating BPMN to executable BPEL4WS.

Another advantage of having a homogeneous and precise representation for processes is that we can deal much more easily with the execution side. The usual situation is that we have a process description on one side, expressed in a given DSL and an execution engine on the other side. To deal with this situation we must usually first elicit the implicit DSL on which the execution engine is based and make it explicit within the same metamodelling framework. Then the two DSLs are compared and a mapping may be defined like the one suggested above between BPMN and BPEL4WS. If later a choice different from BPEL4WS is made, only the mapping has to be changed. Moreover, part of this mapping could be reused if the transformation language has suitable properties.

4 Model-based Process Engineering

Moving from 'contemplative' to 'productive' model management means that most of the steps in the software production and maintenance chain may be considered as precisely defined operations on model artifacts.

A model-based workbench may be viewed as a software bus on which a number of multiple service tools may be plugged with standard interfaces and protocols. The new idea in this organization is that exchanged artifacts are models, now considered as first-class elements.

As an illustration of the possibilities of such an organization, we provide the following example which shows how metamodelling and model transformation may be used for integrating different tools, and particularly legacy tools which do not belong to the MDA technological space.

Microsoft-Project is a widespread planning tool. It is not based on an explicit meta-model. However, its underlying formalism may be extracted and formulated through the metamodel shown in Figure 2.

MS-Project meta-model is based on the concept of project. A project is described as a set of tasks. Tasks are ordered using links. These are assigned to resources.

Creating a project planning from a process model may then be envisioned as a model transformation between SPEM and MS-Project metamodels. A mapping has been established between concepts from both sides (SPEM process and MS-Project project, SPEM work definition and MS-Project task, etc.). Some more complex algorithms have also been designed for reproducing ordering dependencies. Figure 3 illustrates the net result of part of this project as it appears to the end user.

Actually, the transformation was not so straightforward, as the SPEM process is usually not expressed using a SPEM tool, but with a UML modeler extended using a SPEM UML profile (i.e. a lightweight extension of UML as discussed earlier). A transformation from UML to SPEM had first to be applied. The resulting SPEM model could then be used for producing the MS-Project model.

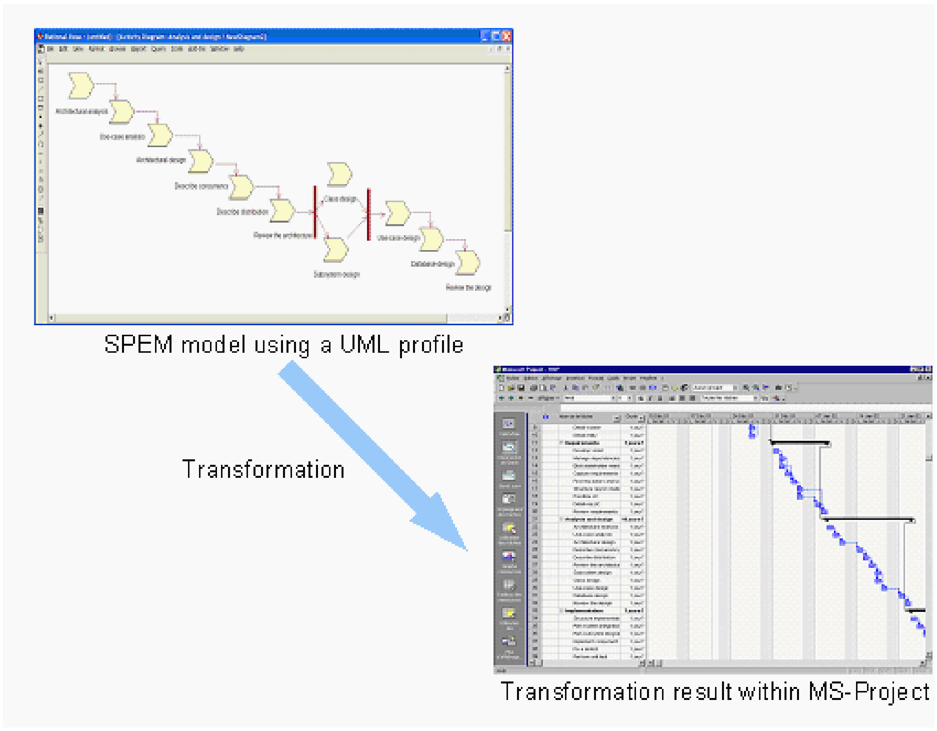

Figure 3: Transformation from SPEM to MS-Project.

Using SPEM as a basis allows the reuse of transformations with different scenarios. A natural way to produce a SPEM process is to use a UML modeler with a SPEM profile. However, there may be some tools producing native (MOF-based) SPEM models. Moreover, we can imagine CAPE (Computer-Assisted Process Engineering) tools with a proprietary formalism. These tools may either propose a SPEM export or a transformation from their proprietary formalism to SPEM has to be provided.

Such an approach has already been used for application management processes in previous projects. From a process model (expressed using a proprietary metamodel because SPEM did not exist at that time), an execution environment was completely generated, including graphical user interface, workflow model and documentation [5].

An important benefit of this approach is that it allows integrating many different points of view through process lifecycle, each point of view being described by a particular meta-model, each transition from one point of view to another by a transformation. Various tools may be integrated since they expose their interface through a meta-model.

5 Conclusion

We have proposed a global method that suggests using model-engineering principles to deal with process definitions. There are several such process DSLs, but they may be related and mapped. Sometimes two process DSLs are similar or one is an extension of another one. In many cases model transformation among model-based process representations has proved to be an interesting idea.

As part of many undergoing projects, the idea of defining a common, open-source MDE platform for various tools is becoming popular. This idea is not new and one may recall projects of the 80's like AD/Cycle (Application Development/Cycle) or PCTE (Portable Common Tool Environment) with similar software buses. However the basic principle that these tools will exchange compatible models, i.e. models conforming to metamodels conforming themselves to common meta-metamodels has proven to be quite powerful in practice. We can envisage a URI-like (Universal Resource Identifier) naming scheme for models like *"model:MMM/MM/M"* where the metamodel *MM* of model *M* is clearly defined as well as the metametamodel *MMM* of *MM*. All the metamodels could be identified in global or local registries called 'megamodels' [3]. A megamodel is a model that contains elements and metadata on the various models, metamodels, services like transformations, tools, etc. For example an explicit semantic relation between the UML and the SPEM metamodel could be expressed in a megamodel. Also the megamodel may contain various metadata on metamodels, models, services, tools, etc. A workflow engine could be registered as a tool for example, referencing its standard model. If we need to execute a process model, we may interrogate the megamodel for a compliant execution engine or we may apply a transformation upon this model. The transformation itself could be applied by another tool linked to the same model-based platform and also described in the megamodel together with its model.

Such scenarios as described in the previous paragraph are becoming realistic in the scope of the present technology. Notice that we do not need to stick to a unique common meta-metamodel as long as the correspondences between the various meta-metamodels are precisely defined. However OMG/MOF and Eclipse/EMF are the two currently aligned industrial dominant standards.

The benefits of such approaches are important. Since everything is a model in this MDE platform, we may include a lot of common facilities like storage and retrieval in model repositories at level M1 and metamodel repositories at level M2, version management, transactional access, etc. Software product and software process models and metamodels will be similarly edited, browsed, stored, retrieved, etc., allowing many economies in tool development cost but also in other areas like user training. We have seen first generation MDE tools (mainly UML tools) with "variable model" capabilities. We are now witnessing the arrival of much powerful tools with "variable metamodel and metametamodel" capabilities. These tools will offer process-engineering or requirement engineering capabilities among others. But what is interesting is that it will be also possible to link to the MDE platform not only MDE tools with extended capabilities, but also legacy tools which were not originally intended to be used in this environment. To integrate such a legacy tool (like MS-Project for example), we have first to define a DSL for the tool, in the form of a standard metamodel like the one of Figure 2, for extraction and injection of proprietary data. From this metamodel, precise guidelines will allow the generation of the injectors/extractors that will enable this tool to be connected to the common MDE platform. From any pair of source/target metamodels we can define syntactic and semantic bridges to convert from one formalism to a similar one or map a process model edited with various kind of editing tool (textual or graphical) onto a given execution engine with a suitable implicit or explicit metamodel.

Model-based process engineering is presently much more than a vision. Based on the three MDE principles (Direct Representation, Automation and Open Standards) and making use of the two basic MDE relations (representation and conformance), it is being deployed in several ongoing industrial projects. However, even if practical problems of product and process model weaving have found ad hoc solutions, more research effort is still needed for a deep understanding of these issues. The development of ambitious open source MDE platform projects hosting simultaneously industrial tools and research prototypes may help to explore still unsolved concrete problems and to improve the state of the art in this field.

Acknowledgements

The work of one of the authors (J. Bézivin) has been supported by a grant from Microsoft.

References

[1]
ATL, ATLAS Transformation Language reference site. <http://www.sciences.univ-nantes.fr/lina/atl/>.

[2]
J. Bézivin. In search of a Basic Principle for Model Driven Engineering, Novatica/Upgrade, Vol. V, N°2, (April 2004), pp. 21–24. <http://www.upgrade-cepis.org/issues/2004/2/upgrade-vol-V-2.html>.

[3]
J. Bézivin, S. Gérard, P. A. Muller, L. Rioux. MDA Components: Challenges and Opportunities, Metamodelling for MDA, First International Workshop, York, UK, (November 2003)., <http://www.cs.york.ac.uk/metamodel4mda/onlineProceedings Final.pdf>.

[4]
G. Booch, A. Brown, S. Iyengar, J. Rumbaugh, B. Selic. The IBM MDA Manifesto The MDA Journal, May 2004. <http://www.bptrends.com/publicationfiles/05-04%20COL%20IBM%20 Manifesto%20-%20Frankel%20-3.pdf>.

[5]
E. Breton. Contribution à la representation de processus par des techniques de méta-modélisation, PhD thesis of the University of Nantes, (June 2002).

[6]
S. Cook. Domain-Specific Modelling and Model Driven Architecture, The MDA Journal, (January 2004. <http://www.bptrends.com/publicationfiles/01-04%20COL%20Dom%20Spec%20 Modeling%20Frankel-Cook.pdf>.

[7]
D. S. Frankell, P. Hayes, E. F. Kendall, D. McGuiness. A Model-Driven Semantic Web Reinforcing Complementary Stengths, The MDA Journal, (July 2004). <http://www.bptrends.com/publicationfiles/07%2D04%20COL%20Semantic%20Webs %20%2D%20Frankel%20%2D%20et%20al%2Epdf>.

[8]
J. Greenfield, K. Short. Software factories Assembling Applications with Patterns, Models, Frameworks and Tools. OOPSLA'03, Anaheim, Ca. <http://www.softmetaware.com/oopsla2003/mda-workshop.html>.

[9]
MIA-Software MIA-Transformation Tutorial. <http://www.model-in-action.fr/download/transformation/3.2.0/tutorial_en.pdf>.

[10]
L. Osterweil. Software processes are software too. ICSE'87, Monterey, Ca.

[11]
R. Soley & the OMG staff MDA, Model-Driven Architecture, (November 2000). <http://www.omg.org/mda/presentations.htm>.

[12]
S. A. White. Introduction to BPMN BPTrends, (July 2004). <http://www.bptrends.com/>.