Using Meta-Model Technologies to Organize Functionalities for Active System Schemes

Erwan Breton
Société Soft-Maint
4, rue du Château de l'Eraudière,BP 588
44074 Nantes cedex 3, France
ebreton@sodifrance.fr

Jean Bézivin
LRSG, Université de Nantes
2, rue de la Houssinière,BP 92208
44322 Nantes cedex 3, France
Jean.Bezivin@sciences.univ-nantes.fr

ABSTRACT

A new information system landscape is emerging that will be more model-centered than object-oriented, characterized by many models of low granularity and high abstraction, which describe static or dynamic aspects of enterprise information systems. This paper focuses more particularly on the organization of dynamic models that we call here active system frameworks (process models by opposition to product models). They cover, among other concerns, the organization of work, its assignment to potential performers and interactions between them. Agent technology is thus concerned in two ways. First, as active elements, they may intervene in a workflow in the same way as real persons. Second, some of their aspects (organization, processing and interaction) are also shared by other domains. All these schemes could and should be organized in a lattice of meta-models. The point of view taken here is essentially the one of the software engineering community. As we witness a rapid proliferation of product and process meta-models, the question of their overall organization arises. Inspiration may be provided in this field by some results of ontology engineering. We specifically point out at two practical areas of concern. First how to couple the meta-models of process and the meta-models of products. Second how to organize the various process meta-models involved in the design and maintenance of information systems.

Categories and Subject Descriptors

D.3.3 [**Software Engineering**]: Design Tools and Techniques – *Object-Oriented Design Methods.*

General Terms

Design

Keywords

Process Models and Meta-models; MOF; MDA; UML;

1. INTRODUCTION

This paper focuses on the particular challenge of interoperability of process models. In section 2 we discuss the new MDA scheme (Model-Driven Architecture) [10] put forward by the OMG (Object Management Group) and the central role of models and meta-models. We show in section 3 how heterogeneous meta-models are interleaved. This may oblige a particular domain meta-model to respect constraints in order to be integrated into more global environments. As we observe similarities between them, we attempt to organize them in a rational way. We particularly study two problems of immediate interest to software engineering. First how to couple the meta-models of process with the meta-models of products. Second how to organize the various process meta-models involved in the design and maintenance of information systems. These problems may be considered as two application areas of agent technology.

2. MODEL DRIVEN ARCHITECTURE

The arrival to maturity of object technologies has allowed the idea of model-based software development to find its way to practicality. The OMG is no more centering its activities on a unique interoperability bus, but on two different ones: the classical CORBA software bus (for code interoperability) and the emerging MOF (Meta-Object Facility) [7] knowledge bus (for model interoperability). The consensus on UML (Unified Modeling Language) [8] has been instrumental in this transition from code-oriented to model-oriented software production techniques. A key role is now played by the concept of meta-model in new software organizations like the OMG meta-model stack architecture. The notion of a meta-model is strongly related to the notion of ontology, used in knowledge representation communities. The concept of a MOF (Meta-Object Facility) has progressively emerged in the last ten years from the work of different communities like CDIF, IRDS, etc., as a framework to define and use meta-models. The MOF is some kind of an upper-level ontology, defined and used by OMG, for the specific task of building information system. Delimiting with precision the boundaries of this task is not easy since it may encompass the definition of business processes for any kind of organization.

We face today a multiplicity of models. The information engineer or the software engineer are usually working with several different ones at the same time, i.e. with models of different semantics. The

executable source code is no more the main and central reference. Product models are arranged in complex organization networks. They are produced and used within precise methodological frameworks sometimes themselves defined by other models (process models). One view of Computer Science is that we are now achieving the first historical paradigm shift, namely from procedural refinement to object composition and that we are, at the same time, starting the second paradigm shift, from object composition to model transformation. Obviously to make this last scheme workable, we need to start with some basic models, mainly the domain model, the resource model and the requirement model. Each model is itself structured in a number of sub-models, allowing for example to define, within the domain model, different entities like business objects, business processes and business rules.

There are two broad families of models: product and processes. A product model may describe for example the software artifacts one may find in an object-oriented design; these artifacts may be described with one of the nine basic diagrams of the UML formalism. Another example of a product model would be the QoS attributes associated with the functional attributes of a similar system. One corresponding process model could be the software process intended to produce the design model or the QoS model.

The decision not to go for a unified object-oriented software production method was initially a wise decision. This conducted to giving the priority to the definition of a meta-model for software artifacts, namely the UML which is basically a product meta-model. The missing link to put this into practice was then the definition of a corresponding generic process and this is what is being achieved with the Unified Process Model (UPM), considered either as a first-class meta-model or as a profile of the UML meta-model. UPM is supposed to provide the building blocks to define a software production process in various environments. At the same time it was also recognized that UML was not sufficient to cover all software artifacts and this is one reason why the CWM (Common Warehouse Meta-model) was defined, opening the path for the definition of broader process like migration processes from legacy systems to advanced component based systems.

All these considerations have brought the OMG to make an important move from OMA (Object Management Architecture) to MDA (Model Driven Architecture) [2], [10]. One particularity of this scheme is the importance given to middleware parameterized model management in the new vision. This means that, as new middleware platforms are emerging and co-existing (CORBA, Java/EJB, C#/DotNet, Web/HTTP, etc.), means should be provided to generate from the abstract models themselves, executable systems for the various target platforms.

3. ORGANIZATION OF META-MODELS

A meta-model defines a language for describing a specific domain of interest. For example UML describes the artifacts of an object-oriented software system. Some other meta-models address domains like process, organization, quality of service, etc. Their number may be very important as identified domains are highly specialized. Though they are first defined as separated components, it is clear that many relationships exist between them. We found two areas as being of urgent concern: firstly how to organize the coupling of meta-models of products with meta-models of processes. Second how to organize the various models of process that we have to manage.

3.1 Coupling of Processes and Products

An enterprise has some objectives, which may be achieved by products or processes. Products are the results of processes. Processes may be performed both by persons of the organization and software components. Finally a process may generate some costs depending on applications and people. Processes may be of different types. For example a bank is concerned with business processes as cash retrieval or account opening. The software development team of a bank performs software processes and produces software components, which are the result products of software processes. The agents who perform the business processes will use these software components as applications.

One of the first problems is to define the precise relations between the various kinds of elements of a product model with the various kinds of elements of a process model. The difficulty here is often to organize the product model in separate chunks that are logically related. For example the UML meta-model defines nine different kinds of diagrams and a given engineer, with particular skills and responsibilities, may for example accept use cases on input and produce class diagrams on output. Modular decomposition of product models is thus of paramount importance. The various modules defined by the product meta-model may play the role of structured type definition in a classical organization whereas the process meta-models provides the basis for defining the control structures.

If this kind of decomposition is necessary, it is far from being sufficient. The product model defines what is being acted upon. The process model is supposed to state who is doing what, when, how and why. As a consequence the process meta-model also cannot be viewed as a monolithic definition. Instead it should be composed of interrelated chunks of information (Figure 1), related to the actors, the performed tasks, the scheduling of tasks, the conditions for executing these tasks, the cost factors, etc.

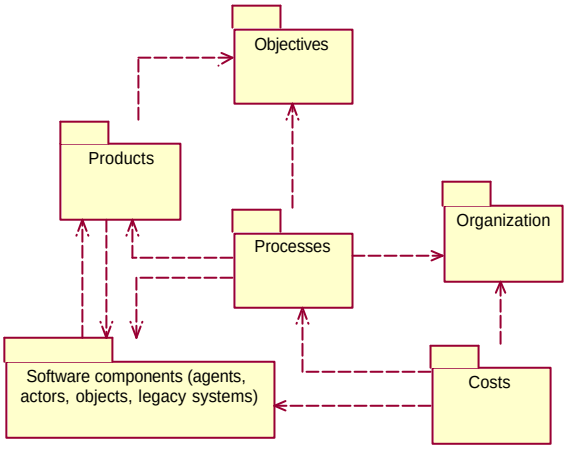

Figure 1 A lattice of product and process meta-models

Agents may be considered (from a process point of view) either as single products or as performers of automated tasks. As a consequence it should be possible to consider them according to both dimensions. They may have to perform a task like a person within the organization. As agents may also perform activities, agent and person specialize a common generic performer.

3.2 Organizing Process Meta-models

There are many separate process domains. This implies many separate meta-models. However there exist some core similarities. A process skeleton is a set of ordered activities (or tasks), grouped in order to achieve common global objectives. It may thus be interesting to propose a process meta-model architecture, allowing a

specific process meta-model to be defined using a more general process meta-model. Expected benefits are reuse and facilities for process models translation from one meta-model to another. Figure 2 proposes such architecture based on domain identification and hierarchy. A generic process meta-model may introduce common concepts like activities, objectives, and transitions. More specific meta-models specialize this root meta-model (or a specific one) by adding new relations and entities. Manufacturing meta-model may thus define mechanisms to handle resources consumption, whereas software process introduces iteration.

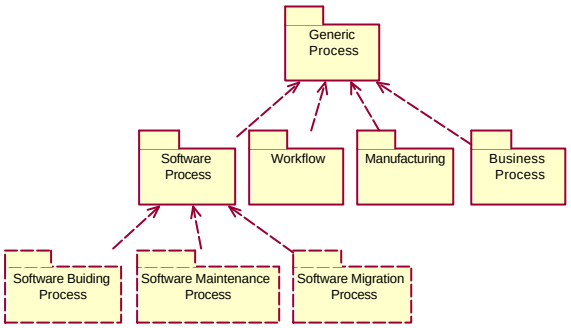

Figure 2 Some meta-models for various processes

Agent technology may also find its place in such an architecture. Business process definition proposals like BPML[1] (Business Process Modeling Language) or ebXML[3] (electronic business XML) define complex interactions between partners, which may encapsulate transactions and activities. The OMG workflow RFP [9] has the purpose to define graph of activities (which may be either manual, automatic or which even may lead to sub-processes). All these meta-models may share a core of common concerns with AUML [3], namely interactions, activity flow and role assignment. It would thus be interesting for some common patterns, or best practices, to be reused across separate domains. In this way meta-models could cover either a particular domain of interest (like agents, business processes or bank documents, etc) or propose standard frameworks (like group organization, interaction between multiple partners, etc.). Framework meta-models would then be reused and specialized by domain meta-models. Domain meta-models would have integration relations between them as they form a strongly interconnected lattice. For example a business process may assign some activities to software agents and others to human beings.

What we may witness today is a lot of separate efforts to define various variants of domain dependent processes like the UPM (Unified Process Model) for production of object-oriented software systems, in conjunction with the UML software product model. Teams working in the related areas of workflow definition, business process specification, maintenance and migration frameworks and many other areas are all encountering similar problems. It would be a wise initiative to put all these contributions together and to study what may be common and what should stay specific. Learning from previous similar projects like PIF [3] should be very helpful in this work. We have today the possibility to perform this meta-model organization in a more operational way, by building on well-accepted industrial frameworks like the OMG MOF. What we need is a global context/organization to host this kind of cooperative work.

4. CONCLUSION

It comes with no surprise that the software engineering community is today discovering the huge potentialities and also some of the unsolved problems of ontology engineering. Once we have laid out the approximation that a meta-model is similar to an ontology, an important work of confrontation remains to be done, with certainly a lot of fruitful outcome. Many insights could be found for example in TOVE [4] or similar research projects. Within a world of increasing complexity due to technological and social evolution, the only sensible way to keep in control is to deal with this complexity at a higher level of abstraction. Many meta-models of products have already been defined to help in this task (UML, CWM, end-user domains meta-models, etc.). It appears that this is only a first step and that the definition of several process meta-models represents also an urgent task (workflow, business process, software production process, software maintenance process, system migration process, etc.). As a consequence we should be in a position to handle, both at the description and at the execution level, the relations between these various process models. Also, the coupling between process models (e.g. workflow models) and product models (e.g. EJB component models) should be handled in a regular way.

5. REFERENCES

[1] BPMI.org **Business Process Modeling Language (BPML) Working draft 0.4** March 2001.

[2] Dsouza, D. **Model-Driven Architecture and Integration: Opportunities and Challenges** Version 1.1. www.kinetiuym.com

[3] ebXML **Business Process Project Team The ebXML Business Process Metamodel Second Draft** June 2000.

[4] Fox, M.S., Gruninger, M., (1997), **On Ontologies and Enterprise Modelling**, *International Conference on Enterprise Integration Modelling Technology 97*, Springer-Verlag, 1997.

[5] Lee J., Grunninger M., Jin Y., Malone T., Tate A., Yost G. & other members of the PIF Working Group **The PIF Process Interchange Format and Framework Version 1.2** The Knowledge Engineering Review, Vol. 13, No. 1, pp. 91-120, Cambridge University Press, March 1998.

[6] Odell, J., Van Dyke Parunak, H., Bauer,B. **Extending UML for Agents** AOIS Workshop, 17th National conference on AI, Austin, TX, pp. 3-17

[7] OMG **Meta Object Facility (MOF) Specification**. OMG Document AD/97-08-14, september 1997.

[8] OMG **OMG Unified Modeling Language Specification version 1.3** OMG Document AD/99-06-08, June 1999.

[9] OMG **Workflow Process Definition Request for Proposal** OMG Document BOM/99-10-03, November 1999.

[10] Soley, R. & the OMG staff. **Model Driven Architecture** OMG Document, November 2000.