An Overview of Industrial Process Meta-Models

Erwan Breton[1, 2], Jean Bézivin[1]

[1] LRSG, University of Nantes,
2, rue de la Houssinière, BP 92208
44322 Nantes cedex 3, France
Jean.Bezivin@sciences.univ-nantes.fr

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

Abstract: The transition from object technology to component technology involves taking into consideration an increasing number of new attributes, representing the multiple facets not only of execution-time, but also of development-time entities. A consequence of this is that the definition of the software process is now becoming a central and critical problem. As a contribution to this area, this paper presents an initial investigation in the organization of meta-models of processes. We have selected several well-known proposals like PIF, NIST's PSL, UPM, WfMC workflow model, etc. We express them as related meta-models, with similar presentation mechanisms based on the OMG recommendations: UML (Unified Modeling Language) and MOF (Meta-Object Facility). We observe some similarities, redundancies and incompatibilities between these models. We conclude this work by proposing a hierarchical organization of process meta-models. The two main questions we wish to answer in this preliminary study are the identification of necessary relations between process meta-models and the hypothetical existence of a core process meta-model. The conclusion will discuss some benefits that can be reaped from a regular organization of process models and meta-models, for ensuring the quality of delivered applications, throughout the life cycle.

Keywords: Model engineering; Process engineering; MOF; RTIM; PIF; PSL; UPM; Workflow; Object-Oriented Analysis and Design

1 Introduction

Process management is today a core concern for software engineering. It promises a better quality of final products, an optimization of costs and delays, a faster adaptation to the new needs of the market, etc. To express software engineering processes we need a basis framework, a software process meta-model. This is the subject of a recent RFP of OMG. Our aim in this paper is not to propose a software engineering process meta-model, but, more humbly, to sketch generalities from the study of various process meta-models. We hope they might be taken in account for the design of a software process meta-model.

As meta-models define a specific viewpoint, a lot of process meta-models have been developed to address a specific domain. Indeed a manufacturing process meta-model and a software process meta-model may have different concerns. For example the manufacturing process meta-model may include real-time aspects, which may be missing in a software process meta-model. Process meta-models may also differ by their objectives. We may distinguish the pure act of modeling organizational behavior (descriptive modeling) from the act of constructing executable systems from these models (active modeling). We are more interested here in descriptive modeling than in active modeling although the boundary is not always very well-drawn between these domains. In any case, descriptive modeling is always a prerequisite to active modeling.

There are important differences between modeling techniques. For example some are fine-grained while other are more coarse-grained. Usually a modeling technique can be associated to an ontology for the purpose of abstraction and sharing. Abstraction means that the ontology defines what should be filtered out from a given situation. Sharing means that there is a collective agreement on the shared meaning of the various concepts (ontological agreement). On practical grounds, an ontology may be assimilated to a meta-model, constituted of conceptual categories and relations between them. To give a naive example, we may consider the simple ontology of Petri nets, composed of the concepts of *Place* and *Transition* together with usual relations between them. Another example in the domain of process modeling could be an actigram IDEF0 ontology.

So, a significant part of the behavioral information captured from a system can be stated in a process model. There is an infinite number of ways for constructing a model of a system, according to the intended purpose of the extracted model. As a consequence, there is also an infinite number of meta-models characterizing these modeling perspectives. We are interested in the practical applicability of process meta-modeling techniques.

The variation in the needs of the modeler are very large and we need to provide adapted techniques. This means that we cannot only offer the choice between Petri nets and IDEF0 actigrams for example. Each process model has its own characteristics. A workflow model is different from an ABC management model (Activity-Based Costing). One of the areas we are mainly interested in, is software process. Even in this area, the differences of perspective are quite important according to the interest, or non-interest, one may place in resource allocation to tasks, in delay measurements, in risk analysis, in consideration of non-functional requirements, and so on.

It is thus becoming clear that, if we want the process model engineering techniques to be practicable, we need to provide a framework to define, combine, use and reuse a set of generic process meta-models. This is the goal we have in the present preliminary investigation.

The rest of the paper is organized as follows. In the next section, we present some state of the art proposals of process meta-models. We use the same notation to picture them, namely the UML notation [11] together with the OCL assertion language [15], in order to make these descriptions more precise. Then, in the following part, we shall be in a position to present some suggestions on the way to organize the hierarchy of meta-models and to use them. The conclusion will summarize the findings of the work and discuss some open problems.

2 Industrial state of the art

This section presents some classical process models. We describe them in a way as uniform as possible, in order to facilitate the synthesis in the following part.

2-1 Early process models: Gantt charts and PERT charts

Gantt charts were created by Henry Gantt in the 1920's. They represent the activities of a process as bars on a calendar. They give a visual representation of the activities, their duration and their planning.

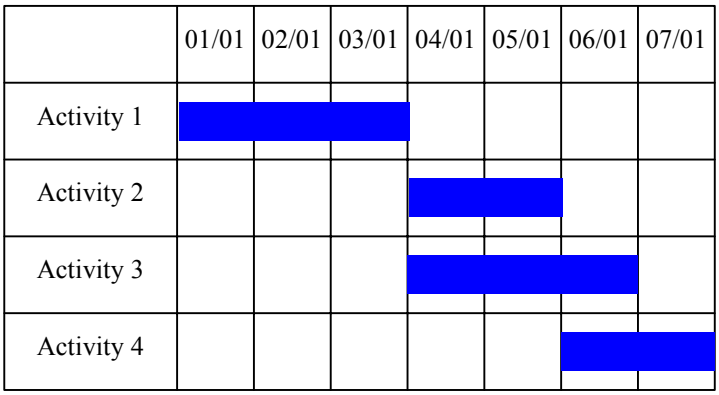

Figure 1: A Gantt chart

As we can immediately infer from Figure 1, the meta-model of original Gantt charts is quite simple as it only defines concepts of activity and timepoint. Some more complex versions exist however and are still very used, for example in project management tools.

PERT (Program Evaluation and Review Technique) charts were first used by the US department of defense. A PERT chart (see Figure 2) is a directed graph that shows tasks, their duration and their precedence relationships. A PERT chart is more difficult to read than a Gantt chart but it allows more complex analysis such as the critical path calculation.

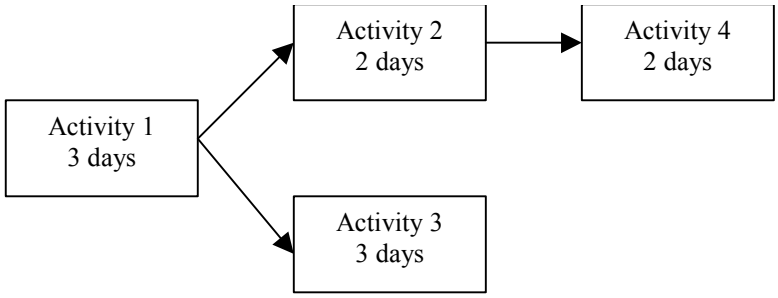

Figure 2: A PERT chart

These two examples will not be examined further but we find it useful to quote them here as they are early tries of process representations, defining minimal set of concepts. They show that there are always been multiple ways of modeling processes, each way using more or less the same concepts but focusing on different properties and thus having its own advantages and disadvantages.

2-2 Process Interchange Format

Process Interchange Format (PIF) [7], [8] stems from the need for various organizations (MIT, DEC, Stanford, etc) to share their process models. PIF specification includes both a process meta-model and a syntax based on KIF. PIF project started in October 1993 and since then the notation has progressively evolved, gaining stability over the years. Today, PSL (see 2-3) and PIF are engaged in a merging process to integrate both business and manufacturing process concepts in a single ontology.

2-2-1 PIF architecture and extension mechanisms

PIF core meta-model defines a more or less minimal set of entities. This basis set may be enriched using the extension mechanism of Partially Shared View (PSV). A PSV module inherits from the core module or from another PSV module (Figure 3). It adds new local entities which specialize other entities defined in the modules on which it is built.

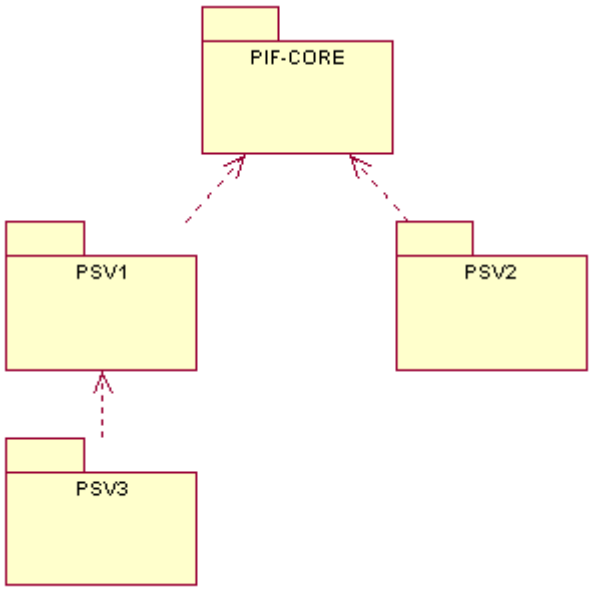

Figure 3: PIF Partially Shared View mechanism

2-2-2 PIF core meta-model

PIF defines a process as "a set of *Activities* that stand in certain *Relations* to one another and to *Objects* over *Timepoints*". These entities, as every other PIF entities, inherit from *Entity* (Figure 4).

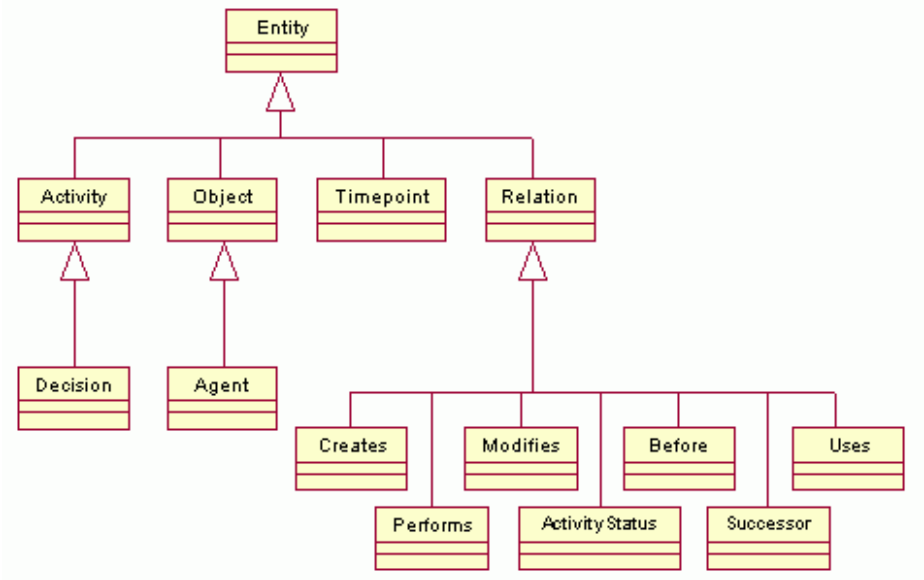

Figure 4: PIF entity hierarchy

Activity is defined as "anything that happens over time", i.e. a process, a task or even an event. *Relation* stands for relations between entities, e.g. *Creates* which links an *Activity* with the *Objects* it produces (Figure 5). *Time-points* may be either a precise hour or a point in time when an event occurs. And finally *Object* represents all other entities involved in a process as artifacts, tools and human or mechanic *Activity* performer (*Agent*).

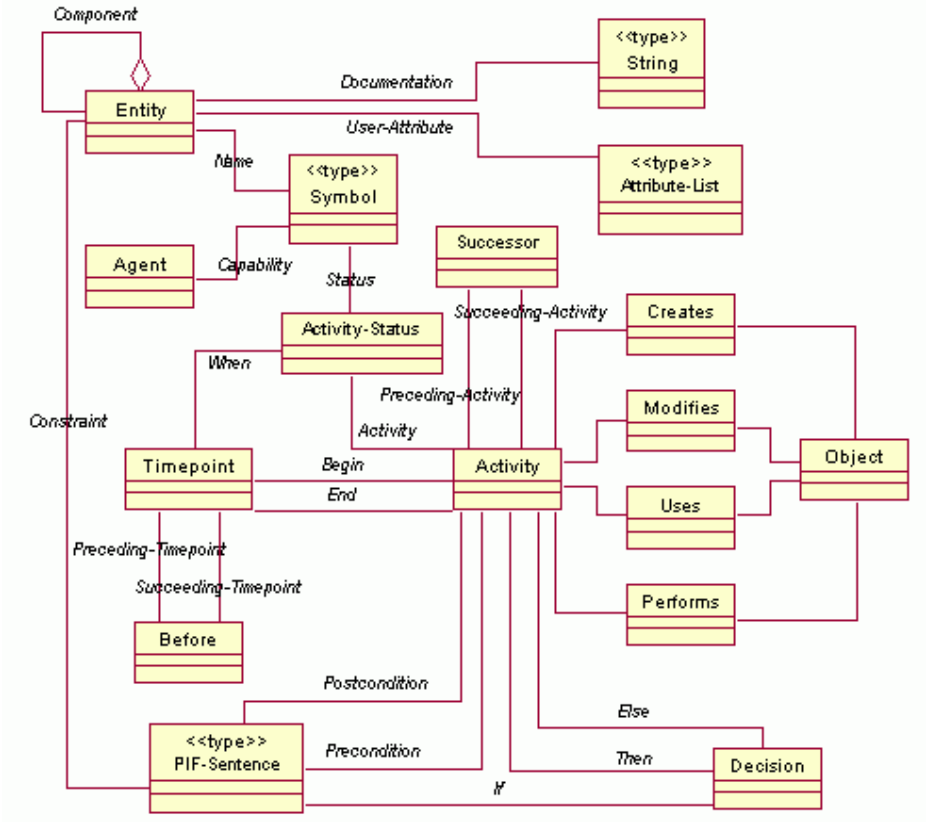

Figure 5: PIF-CORE meta-model

2-3 Process Specification Language

Process Specification Language (PSL, [14]) aims at defining standard ontology and format for exchanging manufacturing processes.

2-3-1 PSL architecture and extension mechanisms

PSL defines a core module based on foundational theories which defines the basic concepts. These are refined in a set of extension module. Some predefined extension modules already exist (Figure 6), each refining a single entity (e.g. the extension "Processor Actions" refines *activity*, the extension "Resource Pools" refines *object*, etc).

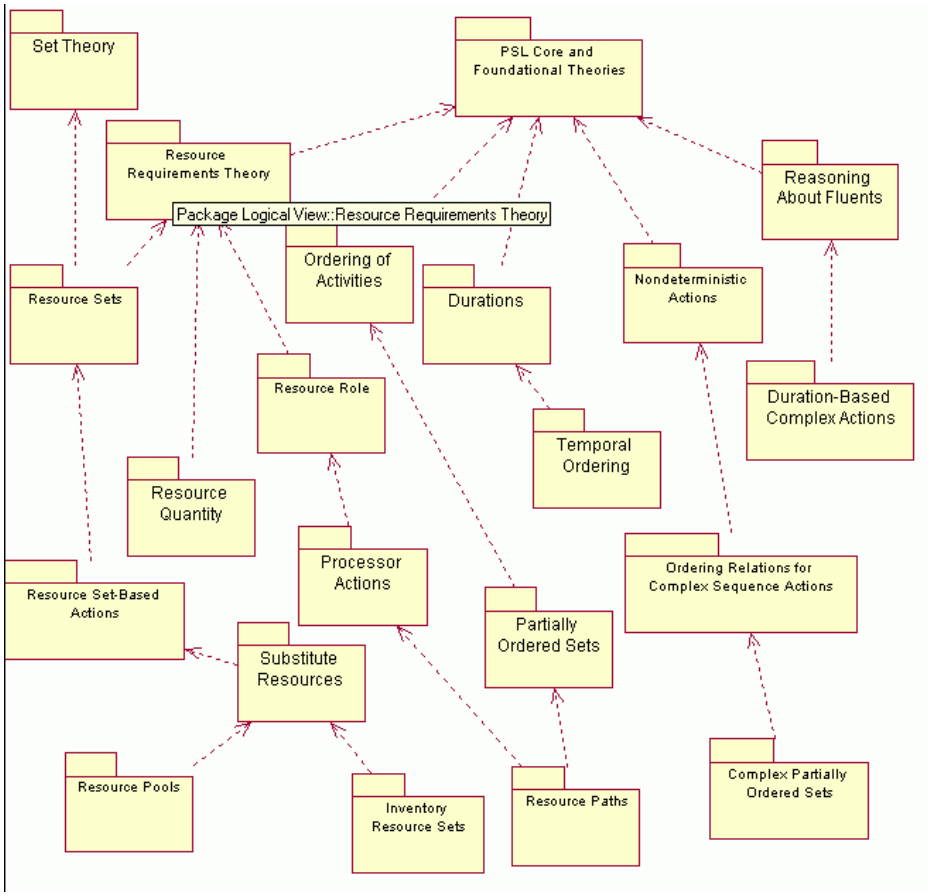

Figure 6: PSL modular architecture

2-3-2 PSL core meta-model

PSL core ontology defines a process as a set of *activity* in which participate some *objects* at *timepoint* (Figure 7). In PSL, everything is either an *activity*, an *object* or a *timepoint*. PSL core also introduces the concept of *activity _occurrence*. This minimal basis is refined in extension modules, defining for example non-deterministic *activities*, quantity for *objects* or temporal ordering on *timepoints*.

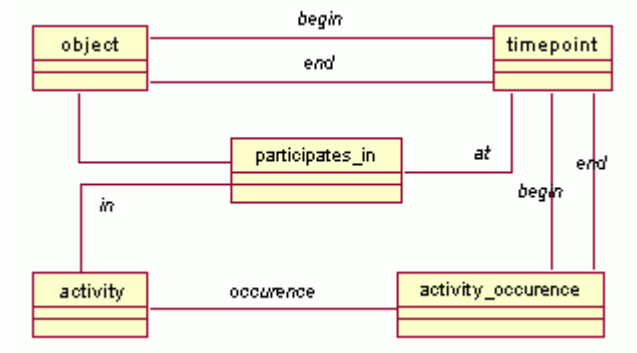

Figure 7: PSL core meta-model

PSL is defined in a very precise and unambiguous manner by so-called axioms or definitions using KIF. Some of these specifications can be expressed in the meta-model, others need to be a more powerful mechanism like OCL.

2-4 Unified Process Model

Unified Process Model is a response to the OMG's Software Process Engineering Management RFP [3], [6]. This specification is a joint proposal from firms like IBM, Rational, Unisys, etc. This process meta-model is already used to define the Rational Unified Process (RUP), a software engineering process model marketed by Rational Software [5].

2-4-1 UPM architecture and extension mechanisms

UPM metamodel includes six packages (see Figure 8):

- *Names* defines naming mechanisms.
- *Basic Elements* defines the basic elements, which are refined in further packages.
- *Process Structure* defines the major process concepts, such as artifacts, role or work items.
- *Process Lifecycle* defines the process execution rules.
- *Guidance* defines how each process component may be documented.
- *Process Components* defines packaging mechanism.

UPM does not currently define any extension mechanism.

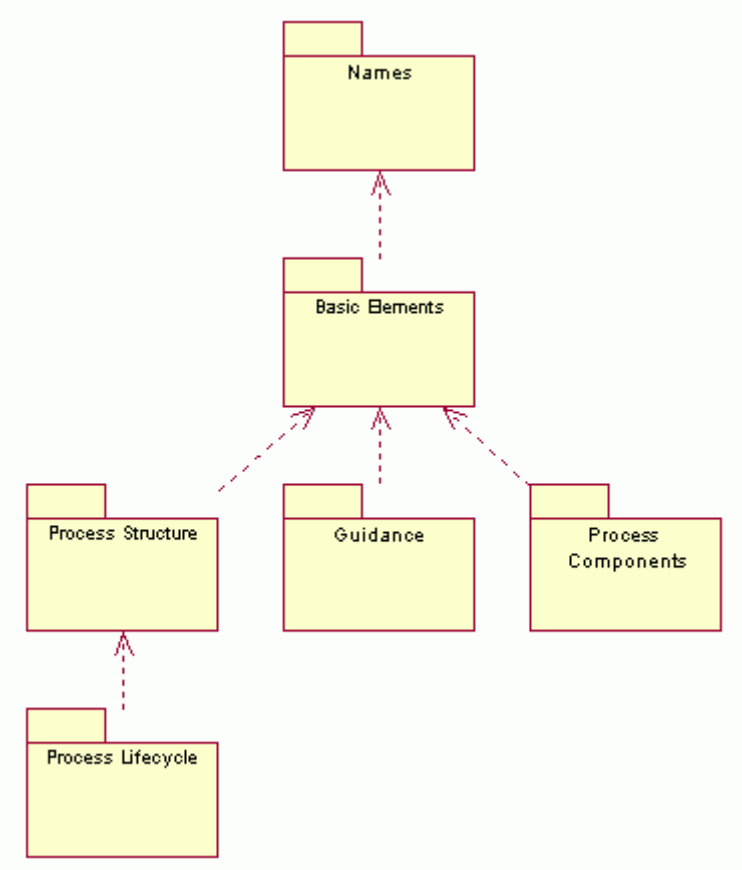

Figure 8: UPM packages

2-4-2 UPM core meta-model

The top-level element of UPM is the *ProcessDefinitionElement* (see Figure 9) which is specialized by most process concepts defined in the meta-model. A *ProcessDefinitionElement* may be documented by a *Guidance*, which is a support of any sort (instructions, checklist, tool guides, etc). *ProcessDefinitionElements* may be organized in coherent sets by using *ProcessComponents*. These mechanisms allow to define end-to-end processes and libraries of reusable components.

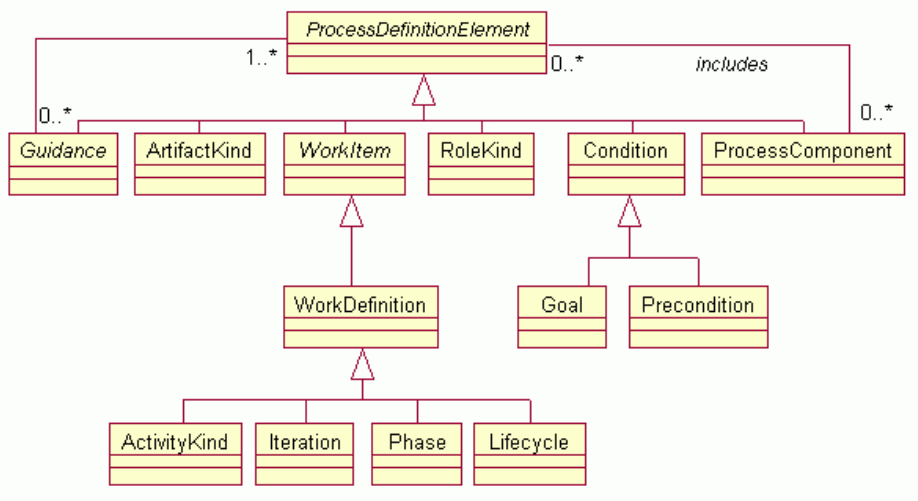

Figure 9: UPM hierarchy

A process is mainly defined by three basic concepts (see Figure 10) :

- *ActivityKind*, a piece of work which may have a *Goal* and a *Precondition*.
- *RoleKind*, a role which may perform or assist in activities and which is responsible of a set of artifacts. A role may be assigned to a person or to a group of people at the execution-time.

ArtifactKind, which represents every kind of information being produced or consumed in a process. An artifact may be an input and/or an output for a specific activity. An artifact may be associated with a set of states, allowing to express *Conditions*.

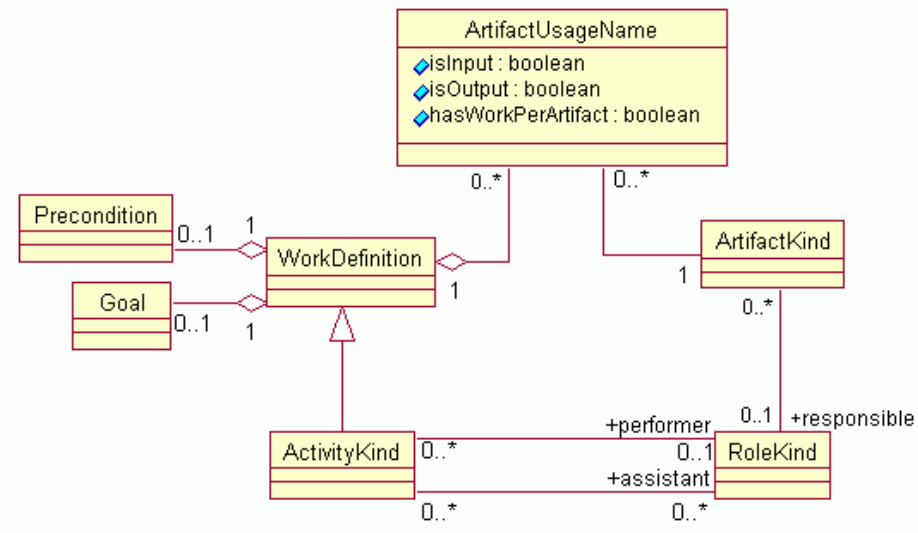

Figure 10: UPM relationships

Lifecycle, *Phase* and *Iteration* define the process lifecycle. The *Lifecycle* of a process is a sequence of *Phases*, each *Phase* being decomposed in *Iterations*.

2-5 Core Plan Representation

Core Plan Representation (CPR, [12]) is sponsored by DARPA and concentrates on planning (specification of a set of actions in order to meet a set of goals or objectives) and scheduling (specification of the amounts of resources used over time and time at which actions will take place).

2-5-1 CPR architecture and extension mechanism

The only extension mechanism of CPR is specialization, new entities being created by specializing basic entities. By this way the core meta-model may be enriched to suit a given domain. Such extensions have already been made for military purposes.

2-5-2 CPR core meta-model

CPR aims at modeling a plan, i.e. a set of actions performed to fulfill some objectives. CPR concepts are *Action*, *Actor*, *Objective* and *Resource* (Figure 11). An *Action* is performed by an *Actor* in order to accomplish some *Objectives*. Performing an *Action*, an *Actor* may use some *Resources*. The *Actor* of one *Action* may be the *Resource* of an other one.

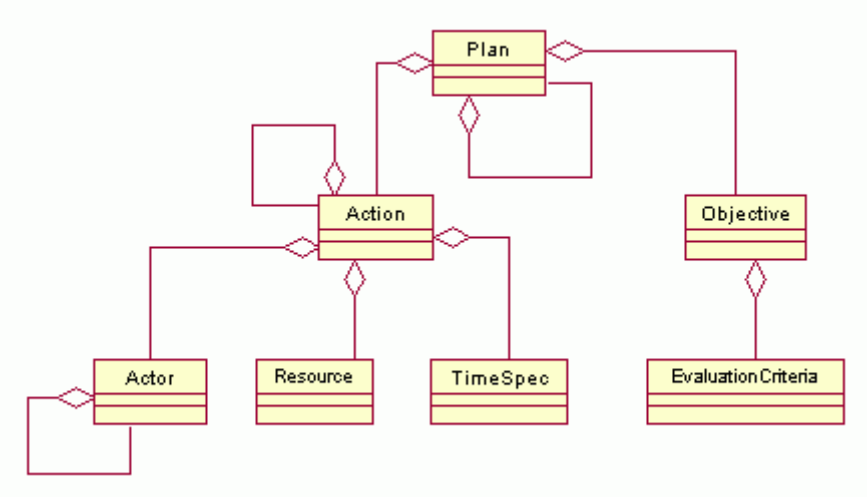

Figure 11: CPR meta-model

This meta-model (Figure 11) allows to model a design plan, as we expect it to occur but it does not allow to record information about how this plan actually occurs. The concept of *WorldModel* (Figure 12) has thus been added to capture execution plans. An execution plan is structured like a design plan but as a design plan foresees for example the amount of *Resources* to be used, the execution plan records the actual amount of *Resources* being used.

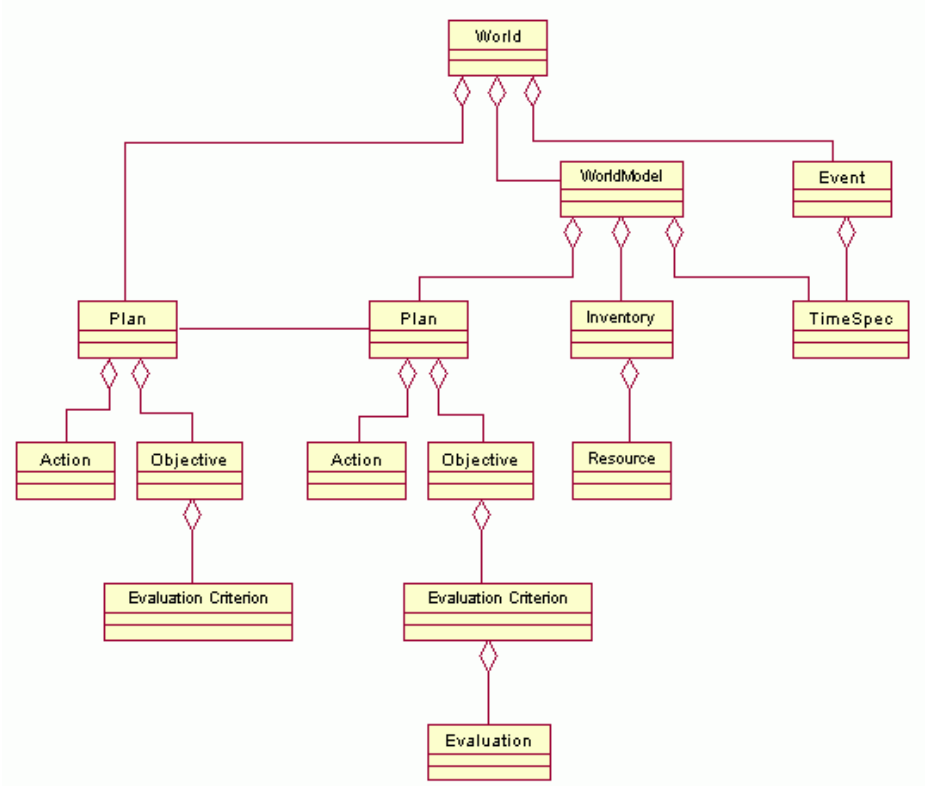

Figure 12: CPR dynamic and static dimensions

2-6 Workflow Management Coalition Process Definition

Workflow Management Coalition (WfMC) is an international organization of workflow vendors, users, analysts and research groups aiming at promoting the use of workflow. It proposes a workflow reference model [16] which defines the following process meta-model.

2-6-1 WfMC Process Definition Architecture

WfMC Process Definition is defined in a single meta-model. However potential relations with external meta-models have been envisioned. For example an organizational model may be linked to the core meta-model in order to enable more complex expressions in the assignment of the performer.

2-6-2 The core meta- model

The process definition meta-model defined by WfMC (Figure 13) concentrates on execution more than on analysis.

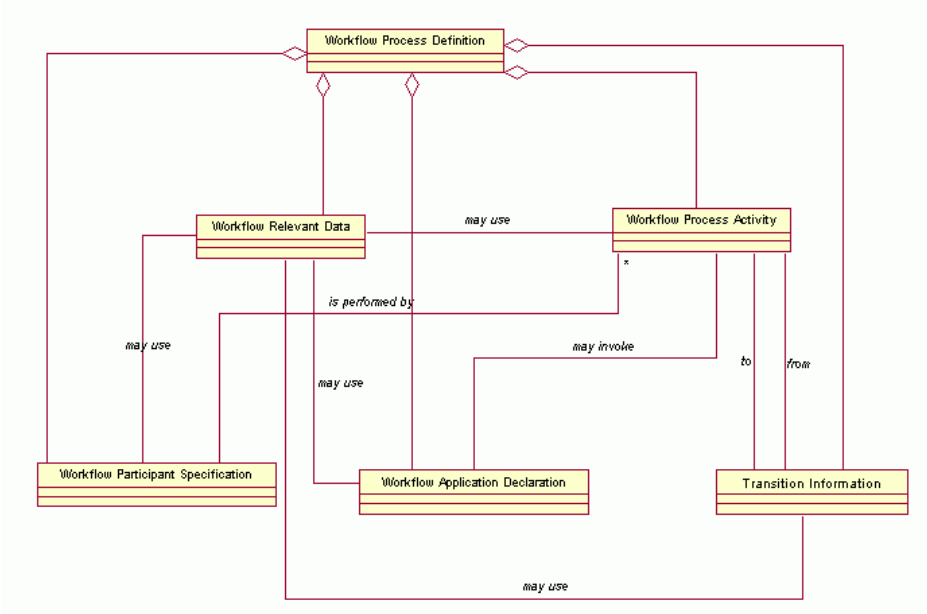

Figure 13: WfMC Basic Process Definition Meta-model

The *Workflow Process Definition* represents the whole process. It is divided in *Workflow Process Activities* which may be atomic or may call a sub-process. These *Workflow Process Activities* are scheduled by *Transition Information* on *WorkflowRelevant Data*. They are performed by someone (or something) defined by a *Workflow Participant Specification* and they may invoke some *Workflow Application Declaration*.

The *Workflow Relevant Data* concept cover a subset of application domain data needed by the process specify transition conditions or activity pre- and post-conditions.

2-7 Architecture of Integrated Information Systems

Architecture of Integrated Information Systems (ARIS, [13]) aims at modeling, analyzing and reengineering business processes.

2-7-1 ARIS architecture

ARIS architecture is divided in five modules (Figure 14). Four of them are independent (Data view, Function view, Organization view and Output view) and introduce thematic sets of entities. The last one (Control view) define relations between entities from different modules.

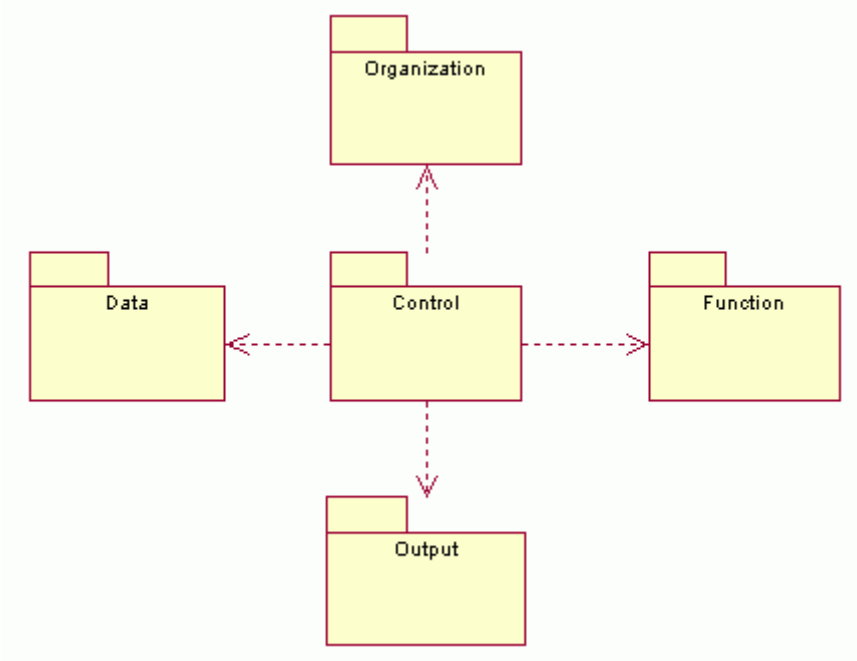

Figure 14: ARIS views architecture

2-7-2 ARIS core meta-model

A process is a set of *Functions* aiming at fulfilling some *Corporate Goals*. A *Function* produces some *Outputs* and processes *Information Objects* such as events or messages. It is performed by *Organizational Units* which cover machines, computers and human resources.

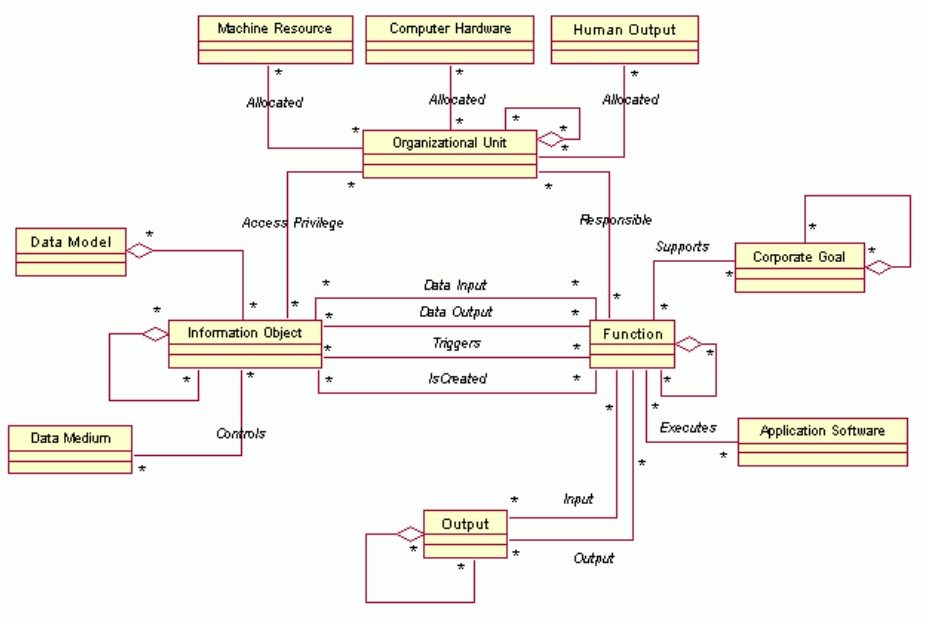

Figure 15: ARIS meta-model

3 Towards a standard framework for process meta-models

Now that we have described a number of proposals, we are going to sketch a general organization for descriptive modeling of processes.

3-1 The three-layer framework

First, we shall follow the now classical three-layer framework[1] that has been proposed in such contexts as *CDIF*, *IRDS*, the *OMG MOF* [10] or Microsoft *OIM* [9]. This framework considers three separate layers usually referred as M1, M2 and M3. The bottom layer M1 is the "model layer". It is there that we find specific models of some systems, defined in term of a particular language, i.e. a meta-model. In the next upper layer M2, called the meta-model layer, we find a collection of such meta-models, one of these being the *UML* meta-model. Other meta-models may be found in this layer like the workflow meta-model or the software process meta-model. All the meta-models are in principle expressed in term of a common language defined by a so-called "meta-meta-model". The supposedly unique meta-meta-model is found in the third layer M3, the meta-meta-model layer. This meta-meta-model is self-defined.

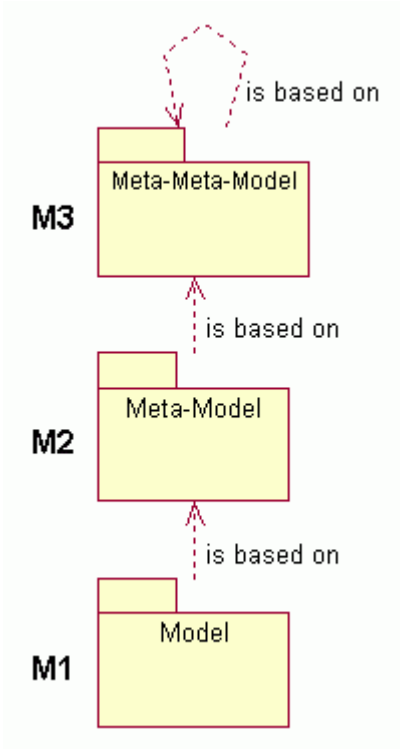

Figure 16: The three-layer architecture

When you are at *OMG*, the third layer is composed of a unique meta-meta-model called the *MOF* (Meta-Object Facility). All the meta-models defined at OMG are *MOF*-compliant, i.e. they are expressed in terms of the *MOF*. All the meta-models are defined at level M2 and the *MOF* is defined at level M3.

[1] As a matter of fact, this is referred in many places as the standard four-layer model architecture. We are omitting here the fourth layer M0 (terminal data), for reasons that are explained elsewhere [[1]]. This problem however does not interfere at all with the subject discussed in the present paper. For our discussion here, we could have kept as well layer M0 and we could have talked about a four-layer framework.

Instead of this, if you are working in a Microsoft context, you should be using the corresponding *OIM* architecture (Open Information Model). The meta-models found at level M2 are called *Uml*, *Umx*, *Dtm*, *Cde*, *Com*, *Gen*, *Dbm*, *Sql*, *Ocl*, *Db2*, *Ifx*, *Tfm*, *Olp*, *Sim*, etc. They are linked in a hierarchy relation (multiple inheritance of meta-models). Of course, the unique meta-meta-model defined at level M3 is not the *MOF*, but *RTIM* (Repository Type Information Model).

The partition between M1, M2 and M3 layers corresponds to different levels of concerns. The M3 layer defines common concepts and mechanisms to all application domains. The objective is to provide a unique formalism, a meta-meta-model, powerful enough to express heterogeneous knowledge. With the M2 layer we go down from a general level to an application-specific level. On the contrary of the M3 layer which is composed by a unique package, the M2 layer is constituted by a myriad of meta-models. Each of them introduces concepts which suit for a particular domain. For example, we could imagine to have a process meta-model and an information system meta-model. The reality is far more complex as we can find manufacturing-oriented process meta-models as well as business-oriented process meta-models or workflow-oriented process meta-models. There is currently no rule to specify what constitutes a real separated domain. There is thus a risk to see the emergence of many closely-related meta-models, only differing by few minor adaptations. Another major concern, and this is a topic we deal in this paper, is the sharing of common concepts and mechanisms between different meta-models. Between the manufacturing-oriented process domain, the business-oriented process domain and the workflow domain , we are convinced that there exists a common set of definitions, like activities, resources, time, etc. This common set may, and must, be extracted and shared by all relevant meta-models. Finally, the M1 layer defines the application level, e.g. a loan request process in a bank or a car manufacturing process in a plant (see Figure 17). Whereas a loan request process should be based on a business-oriented process meta-model, a car manufacturing process should be expressed using a manufacturing-oriented process meta-model.

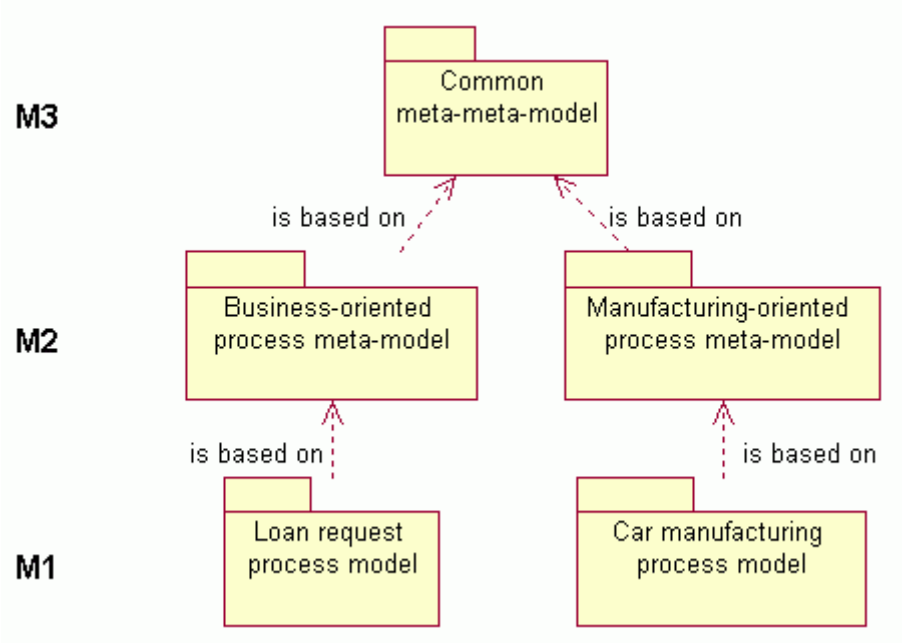

Figure 17: process models and meta-models

3-2 Process and product meta-models

It is now clear that we are going to work at the M2 level. This level is populated by a variety of meta-models. There are two kinds of them: product and process meta-models (see Figure 18). Product meta-models are concentrated on what is produced, i.e. row, intermediate or final information, whereas process meta-models focus on how is manipulated this information, i.e. when, where and by who. Popular (software) product meta-models are

UML, Java, Corba, etc. All meta-models we presented in previous section are process meta-models. It is clear that if you design a process model, the formalism used, which is a process meta-model, is then the product manipulated. This means that product and process are just role, the process specifying how is used, produced or consumed the product.

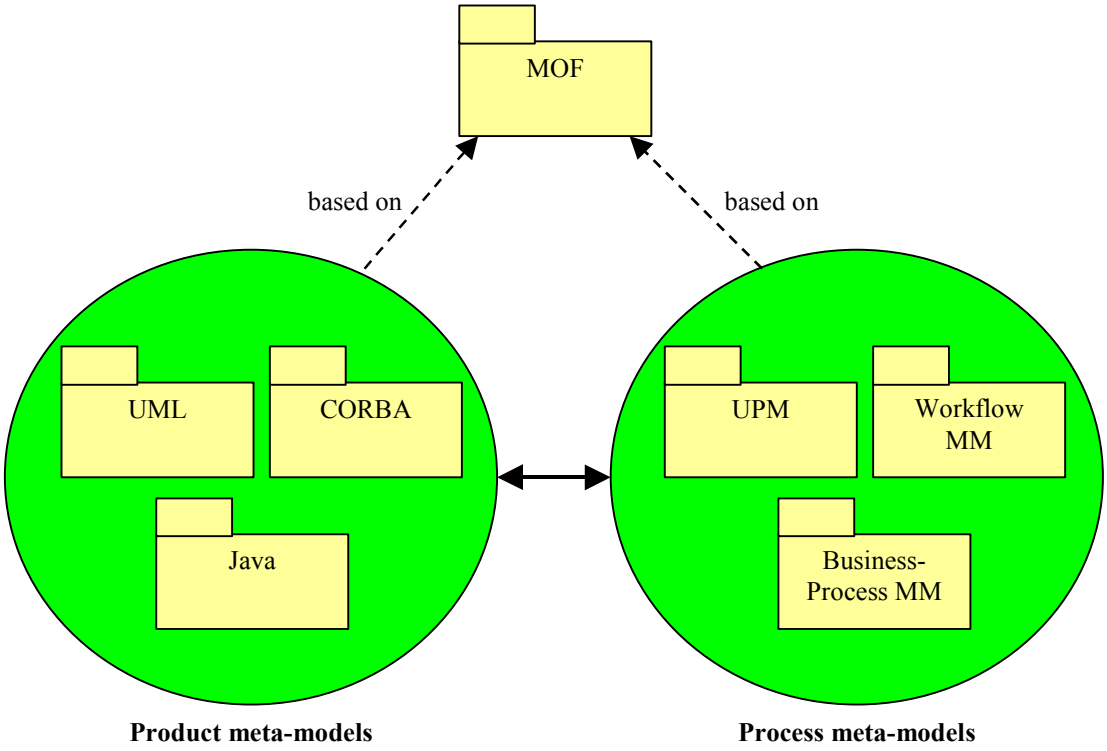

Figure 18: product and process meta-models

Some relations exist between product and process meta-models. For example a pre-condition on an activity may be based on the value of a particular data. This means that product and process meta-models can not be totally independent. With the standardization of the three-layer architecture these process and product meta-models are today expressed using the same meta-model allowing them to be put in relation and comparatively discussed using a common basis [2].

3-3 On the organization of process meta-models.

In the first part of this paper, we mainly focussed on the overall architecture of process meta-models. Some of them are defined in a single package, but, more often, they are divided in several packages. The latter ones may be classified in two different categories. In the first category (see Figure 19) we have meta-models like PIF or PSL which define a highest-level package, introducing the main entities and relations which will be refined in lower-level packages. We call this category “mono-core” as the core meta-model is entirely contained in one and only one package. The second category (see Figure 20) defines a set of highest-level packages which are put in relations in a lower-level package, as it is the case for ARIS. This second category is called “multi-core” as the core meta-model is exploded in several packages. “Mono-core” meta-models stem from an up-down approach, as the overall framework is first created before to be refined in further packages, whereas “multi-core” meta-models are more closed from a bottom-up approach as they initially define independent basic bricks, which will be gathered in a relational sub-package.

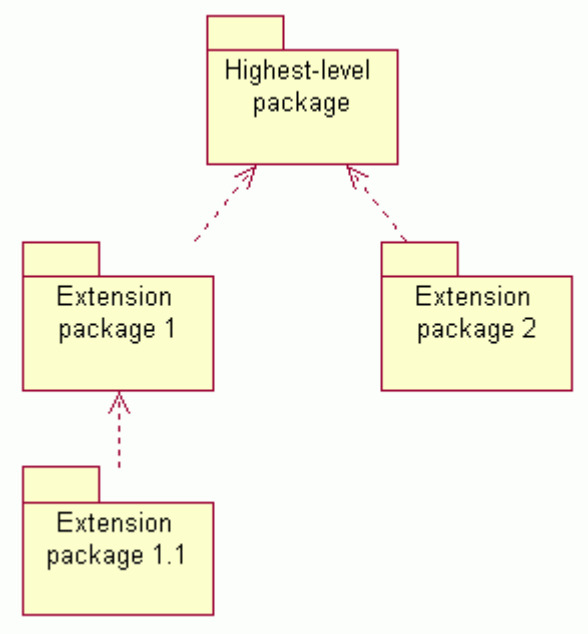

Figure 19: mono-core architecture

In a "mono-core" meta-model, the main concepts and relationships are defined at the same place, in the highest-level package. This implies that the core meta-model comprises a limited set of concepts. This condensing may appear as an advantage as the meta-model is thus very legible. These basic concepts may be refined in sub-level packages to cope with different purposes. For this objective the basic definitions need to be generic enough to allow great variations in their subsequent refinements. A drawback is that some definitions may appear quite vague. And another one, maybe more important, is the risk for an important concept to be missing from the core package. On one hand generic entities like PIF Entity has been defined and this lack could be filled. On the other hand there is no generic high-level entities defined and such lacks could be very serious.

An important advantage of "mono-core" meta-models is the easiness to transform model. If we consider Figure 19, a model based on "Extension package 1.1" may easily be translated in a model based on "Extension package 2". Indeed, as all concepts available in "Extension package 1.1" are either defined in "Highest-level package" or inherits from a concept defined in "Highest-level package", it is quite easy to reduce a model based on "Extension package 1.1" to another based on "Extension package 2" by using the greater common set of entities understandable by both. This is typically what is called PSV (Partially Shared View) mechanism in PIF. It is thus not surprising to find PIF and PSL in this category as their primary goal was to provide exchange facilities for process models.

However, a major drawback of such an approach is that the overall graph of meta-models is supposed to be totally independent from external meta-models. This limitation may appear to be quite restrictive. More and more meta-models will emerge, each new meta-model covering a new domain. It is almost sure that some integration facilities will be needed between them. "Mono-core" meta-models, as they are currently designed, seems to miss this necessity of opening.

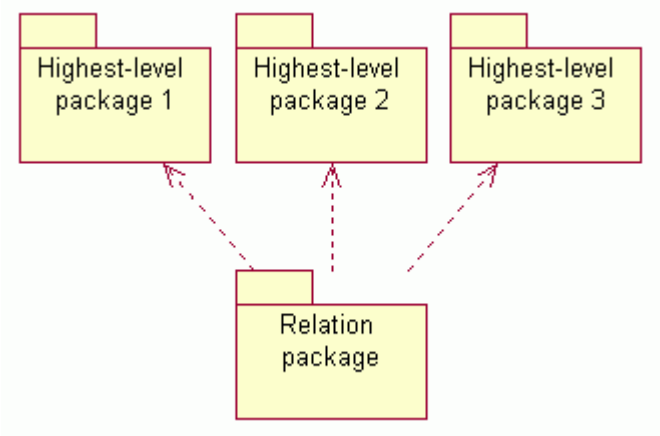

Figure 20: multi-core architecture

The second category, "multi-core" meta-models, does not define a single module but a set of independent modules, each module defining a specific, separated part of the whole meta-model. For example, for a process meta-model, we could imagine a package to be specifically designed for activity scheduling, another for human resources and another for documents. These high-level packages are put in relation by a lower-level package. This lower-level package extends all independent basic modules, and defines relations between elements issued from different modules.

This approach allow to share few sets of basic concepts between heterogeneous meta-models. For example a human resources meta-model could be used both for process modeling and staff management. This may be generalized to integrate popular meta-model like UML with a process meta-model. This could be useful to show which software component will perform an automated task, or to analyze the data flow through a process. A great benefit of "multi-core" meta-models is thus their opening, and their ability to integrate heterogeneous meta-models.

However "multi-core" meta-models have two major drawbacks. First there is a risk of multiplication of relational packages. This would result on an overloaded tree of meta-models, which would be very difficult to understand and to use. And second, the exchange of models could be very difficult between two meta-models based on a different set of basic definition packages.

We feel that the division between "mono-core" and "multi-core" meta-models is destined to vanish. New meta-models will be designed by mixing these two differant approaches. Small specific "mono-core" meta-models will be designed for some very restricted and identified areas. These "mono-core" meta-models will thus be used to be integrated in larger domain-specific "multi-core" meta-model.

3-4 Is there a core process meta-model ?

The basic objective of a process model is to answer these few questions:

- what is to be done ?
- who do it ?
- what is produced ?

Most of the studied process meta-models address these issues, but each in a different way. The following tables compare how some basic notions are represented and what they exactly cover.

- *What has to be performed?*

Meta-model	Concept	Definition	Main relationships
PIF	*Activity*	Anything that happens over time (process, procedure, event).	An *Activity* has some *Succeeding-Activities*. It creates, uses, and modifies *Objects*. It is also performed by an *Object*. It begins and ends at *Timepoints*. At a certain *Timepoint* it may be in a definite state (*Activity-Status*). Finally an *Activity* may have some pre- and post-condition which are *PIF-Sentences*.
PSL	*Activity*	Something that occurs over an interval and one or more objects participate in it.	An *Activity* has some *Objects* which participates in it over *Timepoints*. An *Activity* may be instantiated by *Activity-Occurrences*.
UPM	*ActivityKind*	A piece of work (tasks, operations or actions).	An *ActivityKind* is performed by a *RoleKind*. This performance may be assisted by other *RoleKinds*. An *ActivityKind* has some *Goals* and *Preconditions*. It works on *ArtifactKind* which are known through their *ArtifactUsageName* and which may be input or output.
CPR	*Action*		An *Action* is performed by an *Actor* and has *Objectives*. Some *Resources* may be used during its performance. An *Action* is bounded by *TimeSpec*. Finally an *Action* may be decomposed in other *Actions*.
WfMC	*Workflow Process Activity*	Work which will be processed by a combination of resource and/or computer applications. Activity may be as well as atomic that a sub-process, a loop, or a routing decision.	A *Workflow Process Activity* is part of a *Workflow Process*. It is performed by a participant (defined through *Workflow Participant Specification*). Applications may be invoked through its performance (*Workflow Application Declaration*). It may use *Workflow Relevant Data* and the *Workflow Process Activity* may obey to *Transition Information*.
ARIS	*Function*	The process transforming input in output.	A *Function* supports *Corporate Goals*. It may be triggered by *Information Objects*. It may input and output *Information Objects* as well as *Outputs*. It has a responsible, an *Organizational Unit*, and may execute *Application Software*.

Table 1: comparison of the activity concept

The concept of activity is one of the core concern of every meta-models we studied. However this concept may be slightly different from one meta-model to another.

As there is a consensus for considering an activity as a piece of work, some definition encompasses this simple view. For example PIF integrates events in the activity notion. However, this is quite marginal and there seems to be a global agreement for considering an activity as a piece of a process which may be an atomic operation as well as a complex graph of sub-activities, namely the process itself.

The performance of an activity is defined in almost every meta-model. PSL has not namely defined this relationship, but the “participates_in” relationship between an activity and an object may be used to define this. UPM added a further relationship between an activity and a role, the “assist” relationship which may be used to define that a role may assist in the performance of an activity.

An activity often means a transformation in the universe of product, i.e. a creation, a modification or a consumption. Except CPR, which focuses on military processes which are more concerned with objective fulfillment than product manufacturing, the other meta-models include these concepts. They may be very vague as in PSL where there is only the “participates_in” relationships which is defined, or very significant like in PIF where “creates”, “modifies” and “uses” are explicit. UPM and ARIS introduces “input” and “output” relationships.

Finally some further concepts have been introduced for specific purposes. The objectives are used in high-level meta-model. The process model engineered may thus change as the goals of the enterprise change. This may be quite useful to reengineer a process. Time specifications are also significant for high-level modeling. The process model may thus be simulated and critical path may be evaluated. On the other hand meta-model like workflow meta-model deals with the environment, i.e. the information system and the software applications which may be used. This is quite normal as the purpose is here to produce an executable system integrated in its execution environment.

- *Who do it ?*

Meta-model	Concept	Definition	Main relationships
PIF	*Agent*	A person, group, or other entity (such as computer program) that participates in a process.	An *Agent* is an *Object* that has some specific capabilities to perform *Activities*.
PSL	*Object*	Everything that can participate in an activity.	An *Object* participates in an *Activity* over *Timepoints*.
UPM	*RoleKind*	A role defines responsibilities over a set of artifacts and over a set of activities which it may either perform or assist.	A *RoleKind* may perform *ActivityKinds* or may assist their performance. A *RoleKind* may be responsible of *ArtifactKinds*.
CPR	*Actor*	An Actor performs an Action.	An *Actor* performs *Actions*. An *Actor* may be a Resource for an *Action*.
WfMC	*Workflow Participant Specification*	This provides description of resources that can act as the performer of the various activities in the process definition. It may be a person, a set of people of appropriate skill or responsibility or an automata.	A *Participant* performs *Activities*. The election of a *Participant* may be evaluated using *Workflow Relevant Data*.
ARIS	*Organizational Unit*	Position, department or the enterprise.	An *Organizational Resource* may be responsible of *Functions* and may have *Resources* allocated to it

Table 2: comparison of the actor concept

This concept may be very different from one meta-model to another one. We can distinguish two major categories. In the first one the worker is concrete. It may be a person, a computer program, a department, a position in the enterprise or anything else but it is an entity which exists apart from the process. In the second category the worker is abstract. It defines a role which is played in the process and covers a set of properties (skills, capabilities, degree of responsibility, …) which may be expected from the concrete worker which will be assigned to this role. This notion of abstract worker makes models much more generic and reusable as the role is process contextual, whereas the execution environment (people, teams, tools) is contextual to the enterprise in which the process

is executed. The role may be then used for finding which entity of a specific enterprise is best-suited to perform an activity. A specificity of UPM is that a role may be responsible of artifacts.

- *What is produced ?*

Meta-model	Concept	Definition	Main relationships
PIF	*Object*	An *Entity* that can be used, created or modified by an *Activity*.	An *Object* may be used, created or modified by an *Activity*.
PSL	*Object*	Everything that can participate in an activity.	An *Object* participates in an *Activity* over *Timepoints*.
UPM	*ArtifactKind*	Anything produced, consumed or modified by a process.	An *ArtifactKind* may be in relation with an *ActivityKind* as input or output for this latter. It may be under the responsibility of a *RoleKind*.
CPR			
WfMC			
ARIS	*Output*	Every kind of output (material output, service and information output)	An *Output* may be produced or used by a *Function*.

Table 3: comparison of the product concept

There may be lot of things behind the product concept. On one hand UPM artifacts are considered to be pieces of information. On the other hand ARIS outputs are concrete products like material, service or information. The products may be of different nature according to the nature of the field covered by the meta-model. UPM focuses on software process and ARIS on manufacturing or service supplying processes. There are also meta-models like CPR or WfMC process definition which simply occult the notion of product. As CPR has been designed for military purposes its main concerns are with objectives more than with products. WfMC process definition defines workflow relevant data. These data are a subset of application data which are meaningful for the evaluation on transition condition. Other application data are considered to be invisible from the workflow execution environment.

This tables show that there is no standard name for a concept. More important, two concepts that we may think close have sometimes very different definitions. For example, what is common between a UPM WorkerType grouping properties needed to realize a specific task and an ARIS Organizational Unit which is a position, a department or the enterprise. The heterogeneity of process meta-models may be illustrated by the definition of the process itself which is either an explicit concept, a top-level activity or the whole model. If the common core is the denominator of all these meta-models it would be quite minimal and undefined whereas a more complex meta-model would run the risk to be unable to fit every needs.

3-5 The process modeling language

When defining a process meta-model, a recurrent problem is the definition of the supporting language. This PML (Process Modeling Language) is the format used to store process models and to convey them between tools. For lack of a graphical modeler PML could even be used to design and communicate process models. This is why so many process modeling specifications do not only define a process meta-model, but also PML. For example PIF proposes a PML based on KIF (Knowledge Interchange Format). The problem with this state of fact is that

every new process meta-model use its own language. Thus we have a double level of complexity when comparing two process models issued from two different process meta-models. The first level is due to the difference of content of their respective meta-model. For example a concept used in one model may be not significant for the other one, and vice versa. The second level is relevant to the language in which they are expressed. Before collating their content we must be able to understand both of them. This may be difficult if one model is written using an "exotic", hardly known format.

The emergence of XML (eXtensible Markup Language) is a first step to solve the problem of the variety of PML. XML is now a standard accepted and acknowledged by almost everybody. XML allow to represent information in a simple, readable format, easily parsed by software tools. More than this, there is a real symmetry between a model and its XML file. Particularly, as the model is based on a meta-model, a XML file is based on a DTD (Data Type Definition). A DTD fix the syntax of a XML file by defining a set of tags, each tags corresponding to an entity of the meta-model, the model being thus represented by a XML file (see Figure 21).

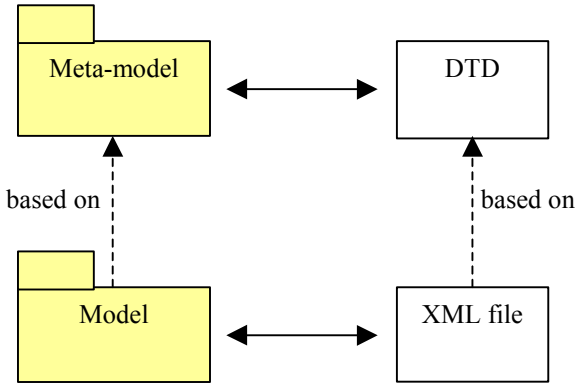

Figure 21: connections between a model and a XML file

XML is thus a first step but is not sufficient as a single meta-model may generate various DTD according to the rules and mechanisms used during this operation. This is the reason why the OMG adopted XMI (XML Metadata Interchange), a standard which reconciles MOF and XML. XMI provides a set of rules which may be automatically applied, and which provides the DTD for any MOF-compliant meta-model. This mechanism allows to feel totally free from designing a PML for the meta-model created as soon as it is integrated in the MOF architecture. This means that the focus is put on the meta-model, i.e. the concepts it defines, and not on its "physical" representation which becomes an appendix of this meta-model.

3-6 Modeling process dynamics

Our main focus in this paper was on descriptive process meta-models. Process meta-models have proved for years their ability to describe processes at a generic static level. Process description may be extracted from a current process in the case of "as-is" modeling, or it may define an expected process if we do "to-be" modeling. Descriptive meta-models are thus particularly well-suited for analysis and design phases where the major concerns are the capture of information (from observations, interviews, statistics and previsions), its formalization and exploration (computation of the critical path, detection of eventual deadlocks). Engineering a process description is always a prerequisite for process evolution, improvement and automated support (or automation). The point is that a process model which only takes into account descriptive aspects can not address such issues. A descriptive model is a passive model [4], it does not evolve in tune with its subject, but stays independent from it once created. It is thus more difficult to compare a process model and its instances for compliance control and process model evolution although process enactment, assessment and evolution are major concerns. In order to address such issues some process meta-models have integrated both static and dynamic aspects. Concepts like process instance (CPR, PSL), state (PIF, UPM) or life-cycle (UPM) have been added to their core. It is quite

important as it promises a seamless integration of the whole process life-cycle with a unique meta-model. But we are still far from this expected result and this issue is still topical.

4 Conclusions

In the first part of the paper, we have analyzed some recent proposals in the domain of process modeling. The notion of a process can take many different aspects, and may involve many different attributes. There is however a commonly accepted idea that the variety of definitions and applications is referring to some common framework, yet to be precisely identified.

The recent events at OMG have seen the quest for a unified object-oriented analysis and design method to be abandoned. Instead, a unified description formalism, covering all aspects of software artifacts, has been accepted under the form of the Unified Modeling Language. The second stage now consists in defining a common software framework, based on the UML notation. It may well be possible that this enterprise could be more complex and challenging than the initial definition of UML. In the present paper we tried to offer a small contribution towards this goal.

One of the new enabling technology is the three-layer model architecture, rooted in a supposedly unique meta-meta-model. This meta-meta-model provide the basic mechanism to define a universal type repository. The use of this technology has recently been boosted by its synergy with standardized transfer syntaxes. The combination of XML with the MOF has given the XMI new OMG standard (XML Model Interchange), while the combination of XML with RTIM has given the XIF at Microsoft (XML Interchange Format).

Of course the large availability of model exchange techniques, including Web-based techniques, does not solve by itself any particular organization problem, even if it gives more credibility to modern model engineering. We mostly need a regular organization of meta-models. The consideration of models (or meta-models, or meta-meta-models), as first class entities is changing the scene of systematic software development. Some general considerations have been presented in this work:

- All product and process meta-models are based on the same meta-meta-model, for example the MOF.
- There are abstract and concrete meta-models.
- Meta-models are related with an extends relation which allows composition.
- A process meta-model should have a *use* relation to all the product meta-models it refers to.
- The hierarchy of process meta-models is somewhat parallel to the hierarchy of product meta-models.
- A process meta-model is constituted by one or more core modules and eventually additional extension modules.
- There is currently no standard core meta-model but all meta-models use more or less the same concepts.
- An important and current topic is the modeling of process dynamics.

5 Acknowledgements

We thank Cédric Pineau that collaborated in the initial part of this work, Jean-Paul Bouchet and Christophe Laurent that are participating in the project and made many interesting suggestions. Last but not least, we thank Jean-Claude Derniame and Philippe Kruchten who offered many comments on this paper.

6 References

[1] Bézivin, J., Lemesle R. **Reflective Modeling Schemes** submitted for publication, February 1999.

[2] Bézivin, J., Bouchet J.P. & Breton E. **Revisiting the P&P Pattern with Explicit Meta-Models** INCOSE/LA Spring Conference, Pasadena, April 1999.

[3] Cook S. (IBM) et al. **The Unified Process Model** AD/2000-05-05, Initial Submission to the OMG's Software Process Engineering Management RFP, May 2000.

[4] Greenwood R.M., Robertson I., Snowdon R.A. & Warboys B.C. **Active Models in Business** Proceedings Business IT Conference, Manchester, 1995.

[5] Kruchten P. **The Rational Unified Process – an introduction** Addison-Wesley, 1999.

[6] Kruchten P. **Unified Process Model (UPM) – A Model of the Rational Unified Process** Proceedings of International Process Technology Workshop, Villard de Lans, France, September 1999.

[7] 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 1.1** May 1996.

[8] 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.

[9] Microsoft, Microsoft Repository Product Information, **Open Information Model Overview**, 1999.

[10] OMG/MOF **Meta Object Facility (MOF) Specification** AD/97-08-14, Object Management Group, Framingham, Mass., September 1997.

[11] OMG/UML **OMG Unified Modeling Language Specification, Version 1.3** June 1999.

[12] Pease A., **Core Plan Representation, version 4** November 1998.

[13] Scheer A.-W. **ARIS – Business Process Frameworks** Springer, 1998.

[14] Schlenoff C., Knutilla A. & Ray S. **A Robust Process Ontology for Manufacturing Systems Integration** Proceedings of 2nd International Conference on Engineering Design and Automation, Maui, Hawaii, August 7-14, 1998.

[15] Warmer J. & Kleppe A. **The Object Constraint Language Precise Modeling with UML** Addison Wesley, October 1998.

[16] Workflow Management Coalition **Interface 1 : Process Definition Interchange Process Model** WfMC TC-1016-P, November 1998.

7 Glossary

ARIS	Architecture of Integrated Information Systems
CDIF	Case Data Interchange Format
CORBA	Common Object Request Broker Architecture
CPR	Core Plan Representation
DARPA	Defense Advanced Research Project Agency
DTD	Data Type Definition
IDL	Interface Definition Language
IRDS	Information Resource Dictionary System
KIF	Knowledge Interchange Format
LRSG	Laboratoire de Recherche en Sciences de Gestion
MIT	Massachusetts Institute of Technology
MOF	Meta-Object Facility
NIST	National Institute of Standards and Technology
OCL	Object Constraint Language
OIM	Open Information System
OMG	Object Management Group
PERT	Program Evaluation and Review Technique
PIF	Process Interchange Format
PML	Process Modeling Language
PSL	Process Specification Language
PSV	Partially Shared View
RTIM	Repository Type Information Model
RUP	Rational Unified Process
UML	Unified Modeling Language
UPM	Unified Process Model
WfMC	Workflow Management Coalition
XIF	XML Interchange Format
XML	eXtensible Markup Language
XMI	XML Metadata Interchange