Weaving Definition and Execution Aspects of Process Meta-Models

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

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

Abstract

The life-cycle of a workflow may be basically summarized to the two following steps: build-time (the process is defined) and run-time (the process definition is used to control and facilitate its enactments). Specific formalisms have been developed for defining processes and controlling their executions. The problem is that these formalisms are separate. The relation between a process definition and its executions is not made explicit. However the behaviour of a process at run-time may impact its definition. There is thus a need for weaving definition and execution aspects. Meta-modeling techniques may bring some relevant answers to this particular issue as they allow the definition of separate aspects of a system and their combination. This is what we attempt to show in this paper by studying the binding between two well-known workflow formalisms (the WfMC process definition meta-model and the OMG Workflow Management Facility).

1. Introduction

Since the adoption of the MOF recommendation (Meta-Object Facility) [10] by the OMG (Object Management Group) in 1997, the importance of model engineering in the information system and the software development process has rapidly increased. The OMG is today no more centering its activities on one bus but on two different interoperability buses. The first one is CORBA, the historical software bus. The second one is the new MOF semantic bus. A key role is now played by the concept of meta-model in new software organizations like the OMG meta-model stack architecture.

At the top of this architecture there is the MOF that provides a language for defining meta-models. It is thus a meta-meta-model. Meta-models aim at describing a particular domain of interest by defining a set of concepts and relations between these concepts. For example UML [11] (Unified Modeling Language) allows describing object-oriented software artefacts. Meta-models make explicit the concepts used for describing a particular domain of interest. The MOF provides a common formalism for defining any kind of meta-model as a set of *MOF::Classes*, *MOF::Associations*, *MOF::Attributes*, etc. Constraints may be added to the basic concepts using OCL (Object Constraint Language). A four-layer architecture has progressively taken shape. It is organized as follows:

- M3: the meta-meta-model level (contains only the MOF).
- M2: the meta-model level (contains any kind of meta-model).
- M1: the model level (contains any model with a corresponding meta-model in M2).
- M0: the concrete level (contains any real situation, unique in space and time, described by a given model).

The emerging OMG MDA [18] (Model Driven Architecture) is a further step towards model-centered software engineering. It is based on OMG modeling standards e.g. the MOF, UML, CWM [9] (Common Warehouse Metamodel) and XMI [15] (XML Metadata Interchange). It aims at providing both an environment for defining platform-independent models (PIM) and generation services to map these models to a particular platform (CORBA, EJB, MTS, etc.) and to produce platform-specific model (PSM). The problems of interoperability are therefore moving from the code level to the model level. In addition to UML and CWM there will be multiple other meta-models. The future OMG workflow process definition (a RFP [11] is currently pending) will naturally find its place within this architecture.

This distinction between PIMs and PSMs may also be applied to the field of workflow. In order to be independent from the execution platform (the workflow management system), the process definition has to be expressed using a formalism neutral from the technology point of view. This is not currently the case. Most workflow management systems propose today their specific formalism and input format. This is quite a

handicap for shifting from one system to a new one, as the business models are "polluted" by platform-oriented concerns. The MDA allows thus preserving business invariants from technology shift by proposing some standard business-specific meta-models.

However, a standard "one size fits all" meta-model does not constitute an acceptable answer to face the increasing complexity and diversity in the needs. Standards constitute a stable basis on which to build more specific meta-models adapted to particular domains. There is thus a risk that all those extensions add new concepts without precisely defining their behaviour, which would make subsequent models difficult to be created and understood. Moreover they could not be simulated, as the execution rules would not be expressed. Therefore we need to have some mechanisms for specifying how a process model has to be interpreted, according to the concepts defined in its meta-model. Holger Giese has already identified some similar issues with UML [5]. He stated that the UML core behavioural semantics must be precisely specified to disable the risk of defining domain-specific extensions whose semantics would be inconsistent.

In this paper we attempt to define, in a consistent manner, concepts for process definition and execution. Our objective is to study how a process meta-model may encompass these two aspects. Such a meta-model would then integrate both the terminology and the rules for interpreting it. As UML defines the concepts of *Class* and *Instance* and their relation, a process meta-model should specify how the execution of a process is related to its definition. This would enable to manipulate models for static reasoning (like the critical path determination) as well as instances for studying the exact behaviour of the process in a particular context.

We support our discussion by showing how two well-known proposals for process definition and execution could be combined. First we present the WfMC (Workflow Management Coalition) process definition meta-model. We compare the precise meta-model with its textual documentation. While the documentation gives many details on the behaviour associated with each concept, the meta-model does not specify any behaviour at all. Next we present the OMG workflow management facility. It defines a minimal set of interfaces for controlling workflow execution, and is therefore centered on the definition of execution. Then we are in position to sketch out how the WfMC meta-model could be completed by execution aspects directly inspired from the OMG specification.

2. The WfMC process definition meta-model

The Workflow Management Coalition (WfMC) is a non-profit, international organization of workflow vendors, customers and users. It aims at producing standards for the implementation of workflow products, in order to enhance their capabilities of interoperability. These specifications are described in five interfaces [7]. We are more particularly interested in the interface 1, the workflow definition interchange, which is described in [20] and [21]. It defines an interface between process modeling tools and workflow enactment services. It is based on both a meta-model and a set of APIs. In this section we attempt to compare the textual description of this meta-model with its precise definition.

A workflow is the automation of a business process defined within a process definition. A process definition defines the elements that make up a workflow: activities, transitions, applications, participants and workflow relevant data. A process definition is identified by a unique process identifier. It has a name, a description, a creation date, an author, a publication status, a version, a classification, etc. A particular participant is responsible for the execution of a process instance. The following state machine (Figure 1) is presented as defining the lifecycle of a process instance.

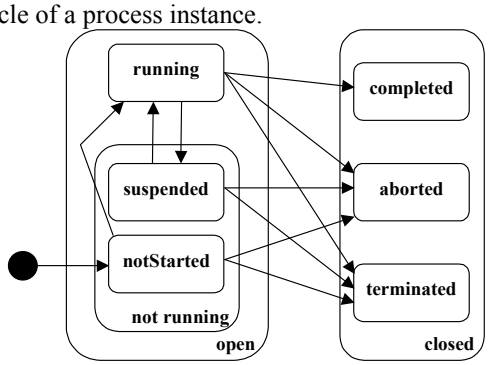

Figure 1. The states of a process instance

The activity definition is used to represent each elementary activity that makes up a workflow process. An activity has an identifier, a name and a description. Expected duration, cost, working and waiting time may be indicated. It has also an implementation type, which may be:

- *None*: it is a manual or a vendor-specific activity.
- *Application*: it is an automated activity that is implemented by tools such as application programs or built-in libraries.
- *Subflow*: it is a call to a sub-process. It may be synchronous or asynchronous. In the first case the execution of the activity is suspended after a process instance of the referenced process definition.
- *Loop*: it is a loop that allows repeating a sub-flow of activities. The implementation of a loop is executed possibly zero times or repeatedly until the loop condition is evaluated to false (*WHILE loop*) or to true (*REPEAT_UNTIL loop*). For the *WHILE loop* the test of

the condition is performed before executing the loop implementation, for the *REPEAT_UNTIL loop* afterwards.

- *Route*: it is a "dummy" activity that has no side effect. It is used for combining XOR- and AND-split conditions on outgoing transitions from an activity, and XOR- and AND- join conditions on incoming transitions to an activity.

Each activity defines how the incoming transitions are joined. In case of a AND-join, synchronisation is required. If it is a XOR-join, the activity is initiated when the transition condition of any of the incoming transition is evaluated to true. A similar mechanism is used for outgoing transitions. A AND-split indicates that all outgoing transitions may be activated (provided their condition evaluates to true), while a XOR-split indicates that only one transition may be activated (the list of outgoing transitions must thus be ordered).

The begin (end) activities of a process definition are those that have no incoming (outgoing) transition. If there are multiple begin activities, they are started concurrently when a process instance starts.

A transition defines a point in the execution when one activity completes and the thread of control passes to another, which starts. A condition may be associated to a transition. A transition is enabled if its condition is evaluated to true.

An application defines an external application that may be invoked to automate an activity, fully or in part.

A participant may be any resource that can be assigned to perform a specific activity, i.e. a person, a set of persons of appropriate skills or responsibility or even a machine. It may refer to an organisational model that enables the evaluation of more complex expression.

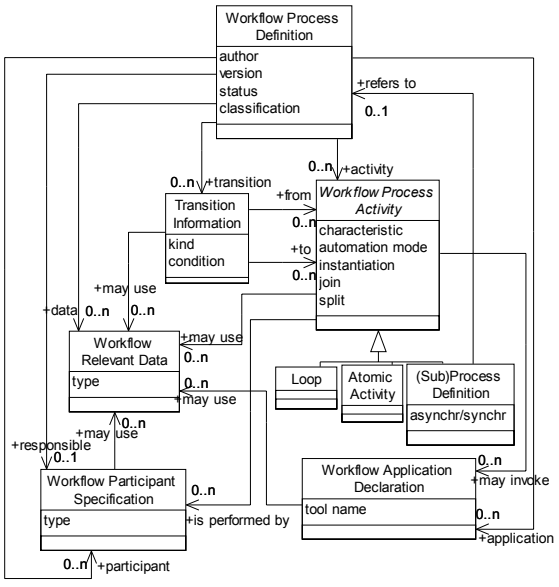

Figure 2. The WfMC definition meta-model

Workflow relevant data are the data created and used within a process instance. They are used to pass results between activities, to record decisions or to evaluate transition conditions.

The WfMC process definition meta-model presented in Figure 2 should be an accurate and exhaustive picture of the textual description given above. The process definition is defined by the *Workflow Process Definition*, the activity by the *Workflow Process Activity*, the transition by the *Transition Information*, the participant by the *Workflow Participant Specification*, the application by the *Workflow Application Declaration* and the workflow relevant data by the *Workflow Relevant Data*. Identified sub-types of activity have also been defined. The *Atomic Activity* covers manual, automated or routing activities, the *Loop* represents both *WHILE* and *REPEAT_UNTIL* loop and the *(Sub)Process Definition* defines a subflow.

We may also find in the meta-model every property we define as attribute. Finally the associations between concepts are also represented. A *Workflow Process Definition* is made up of *Workflow Process Activity*, *Transition Information*, *Workflow Application Declaration*, *Workflow Participant Specification* and *Workflow Relevant Data*. It also has a responsible, which is a *Workflow Participant Specification*. A *Workflow Process Activity* is performed by a *Workflow Participant Specification* and may invoke *Workflow Application Declaration*. A *SubProcess Definition* refers to a *Workflow Process Definition*. A *Transition Information* stands from a set of *Workflow Process Activities* to another set of *Workflow Process Activities*. Finally *Workflow Process Definition*, *Workflow Process Activity*, *Transition Information*, *Workflow Application Declaration* and *Workflow Participant Specification* may use *Workflow Relevant Data*.

Much information that we may extract from the text is thus defined in the meta-model. However many others are missing too. States, events and operations are not defined. For example the following information cannot be found by the exclusive observation of the meta-model:

- The behaviour of the instances of a particular *Workflow Process Definition* is defined by the state machine presented in Figure 1.
- When a workflow instance starts, all its begin activities have to be started concurrently.
- A synchronous *(Sub)Process Definition* waits for the completion of its sub-process.
- Neither join/split mechanism nor loop one is explicit.

The WfMC process definition meta-model defines thus concepts, associations and attributes. It focuses on the definition of concepts and their organization, i.e. how they could stand in relation to each other. However, when looking at this meta-model, we have no information about

the manner a model based on this formalism is executed. And yet the authors of the document recognize that "*since a build time representation of the process definition will subsequently be used to support enactment it is necessary to define a number of basic principles and semantics of the execution time environment*" [20]. All these principles are given either in a textual form or, when precisely defined (see the state machine of process instances presented in Figure 1), they are not related to any element in the meta-model. This meta-model introduces only some definition aspects. It allows describing models but it does not specify anything about the behaviour of the instances of these models. Neither the creation of an instance, nor its evolution through state changes is defined.

3. The OMG workflow management facility

In the former section we have studied the WfMC Process Definition Meta-Model that provides a language for describing process models but does not address execution issues. The OMG workflow management facility specification [14] has different goals. It defines a set of interfaces for "*workflow execution control, monitoring and interoperability between workflows defined and managed independently from each other*". It specifies how a particular execution of a process may be controlled and monitored, in an independent way from any process definition and effective implementation. Therefore it does not precisely describe the behaviour of a process instance. However it defines a framework for defining execution aspects of processes. In this section we study this specification and we discuss the eventual dependencies that may appear between a particular implementation and a process definition meta-model.

A particular workflow is initiated by a requester, which is responsible for that process. It is associated with a process instance when a process instance is created. It is notified of any change in the state of the process. A workflow requester may be either an activity (which is thus a sub-process) that completes once the requested workflow did, or an external application. The workflow is created by a process manager, which represents a template for a specific process definition. It is the factory and locator for workflow instances. The enactment of the process is initiated by invoking the start operation. Begin activities have then to be activated. An activity may be implemented by a sub-process, and so completes once the sub-process has completed. It may also be implemented by an application. In this case the application must explicitly send its results to the activity and completes it. Activities may be assigned to resources (a person or a thing that will potentially accept an assignment to an activity) that participate in the execution of that work. When an activity completes, its results may be used to determine the follow-on activities and the results of the overall process. The process completes when there are no more activities to be activated. Processes and activities evolve from state to state. The identified states and transitions between these states are described in Figure 1. Each state change is recorded by event audits. The creation of a process, the change of a data or the assignment of an activity to a resource are also recorded. Finally some relevant events have also been defined. The model corresponding to this description is presented Figure 3.

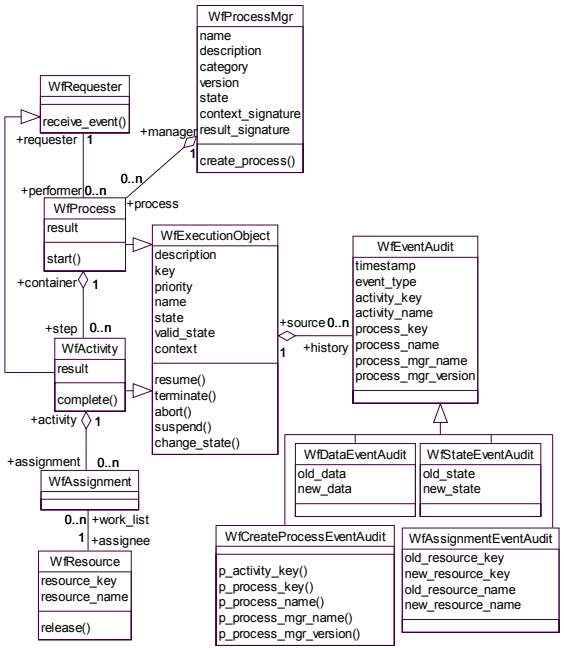

Figure 3. The OMG execution model

The *WfExecutionObject* is the common root for *WfProcess* and *WfActivity*. It manages internal states through the state machine shown in Figure 1. A transition from the current state to another one may be performed using an operation such as *resume*, *terminate*, *abort* or *suspend*. A *WfProcess* performs a workflow request. Its execution is initiated using the start operation. It may contain zero or more *WfActivities*, which are associated with a single *WfProcess*. It may not be explicitly completed as its completion depends on the completion of each *WfActivity*. When a *WfActivity* completes the workflow process determines which activities are open and ready to start or resume. *WfActivities* cannot be explicitly started; their start depends on the process logic. *WfActivities* may be assigned to *WfResources* through *WfAssignments*. A *WfActivity* may be realized by a sub-process. It specializes the *WfRequester* interface, which represents the object that starts the process. This may be a *WfActivity*, a role or an external client. A *WfProcess* is created by a *WfProcessMgr*, which is a template for a specific workflow process. Finally every change is

recorded through a *WfEventAudit* (which may be specialized according to the nature of the change). The meta-model is thus an accurate picture of its textual description.

Comparing the OMG model and the WfMC meta-model we may find some structural similarities. As a *Workflow Process Definition* is made up of *Workflow Process Activity*, a *WfProcess* is composed of *WfActivities*. There are many attributes that seem common between the *WfProcessManager* and the *Workflow Process Definition*. Beyond the *name* and the *description*, they both define a *version*. Moreover, the *category* may be compared to the *classification*, the *state* to the *status* (although the state machines are different), and the *context_signature* may be defined by the set of *Workflow Relevant Data* contained by the process definition.

Finally, the OMG facility integrates aspects that are lacking in the WfMC meta-model. A state machine is explicitly related to process and activity. The relevant stimuli are also specified (i.e. the methods *resume()*, *terminate()*, *abort()*, *suspend()*, etc.). However, this specification only defines a set of interfaces. The exhaustive and precise behaviour has to be defined within implementations. This behaviour may depend on arbitrary choices but it is also strongly guided by the process definition format. For example, the following points need to be made explicit:

- How a *WfProcess* is configured once it has been created using the method *create_process()* of *WfProcessMgr*?
- How a state change (like a suspension or a resumption) is propagated from a *WfProcess* object down to *WfActivity* objects?
- What is the precise logic used to determine which *WfActivities* are to be started or resumed once a *WfActivity* completes?

4. Weaving definition and execution aspects

In the former sections we have studied the WfMC process definition meta-model and the OMG workflow management facility. These two models contain different concepts and have different purposes. The first one introduces every concept needed to describe a workflow model, while the second one specifies how an execution of a workflow may be controlled and monitored. However we saw that interdependencies exist between them. Many questions raised by the WfMC meta-model may be answered by a particular implementation of the OMG specification. In this section we study how this could be done in a consistent manner, i.e. how execution concepts inspired by the OMG specification can be added to the WfMC meta-model.

This section is composed of three distinct parts. First we examine the organization of definition and execution aspects within the MOF meta-models architecture. Once the models are bound we are in position to define associations between definition and execution aspects. And finally we discuss a precise specification of actions.

4.1. Organizing the meta-models

The MOF provides a language for defining meta-models. A meta-model is a set of concepts, associations between these concepts and constraints that describe a particular domain of interest. The meta-model concept itself, i.e. the container, is represented by the *Package* construct. A *Package* may extend other *Packages*, i.e. the concepts defined in the extended *Packages* may be referred from the extending *Package*. The M2-level of the MOF architecture is constituted by a complex lattice of meta-models.

The WfMC process definition meta-model and the OMG workflow management facility have not the same status in this architecture. The definition meta-model is used to define process models. It is thus positioned at M2-level like UML or CWM. The OMG facility is a UML model and therefore it belongs to the M1-level. In this way definition and execution aspects cannot be combined. There is thus no explicit relation between a process definition and its executions (see Figure 4).

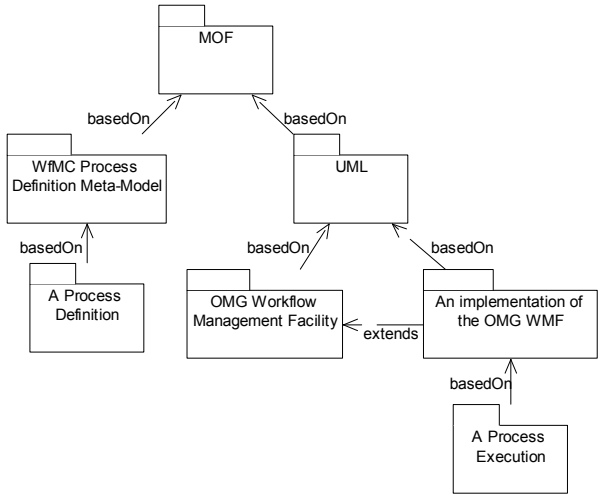

Figure 4. Process definition and execution models

Our purpose is to complete the process definition meta-model by a process execution meta-model. The description of a process definition may be free from any execution concerns. On the other hand the specification of process execution is dependent on a particular process definition. This point has been already raised during the study of the OMG workflow management facility. For this reason the process execution meta-model must extend the process definition meta-model (see Figure 5).

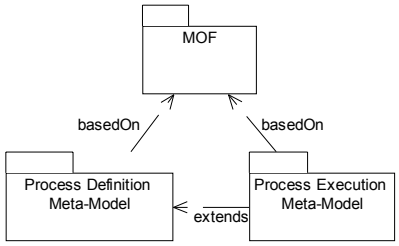

Figure 5. Process definition and execution meta-models

Such an architecture is very flexible. First, it allows defining multiple execution meta-models for a single definition meta-model (see Figure 6). It is frequent that the same definition formalism is interpreted in different manners. In this way the particularities of each interpretation are made explicit. As they are defined in a common language their differences may be easily underlined. Moreover, a new process execution meta-model may be created by adapting an existing one.

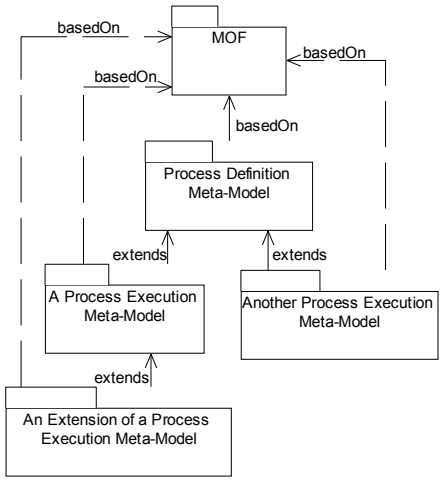

Figure 6. Different process execution meta-models

Second, it allows for extending the process definition and execution meta-models in a consistent manner. For example we may extend the basic WfMC meta-model by introducing in the process definition meta-model a mechanism for representing the "Deferred Choice" [19]. A deferred choice may be compared to the XOR-split: one of several branches must be chosen. The main difference is that the XOR-split defines the choice criteria within the model, while the deferred choice is explicitly made by the environment (i.e. by a participant) during the execution. All branches available are proposed, but only one can be elected. The others must then be withdrawn. An extension of the WfMC meta-model defining such mechanism would no more be consistent with the standard execution meta-model. This latter must be extended too in order to assign a behaviour to the additional concepts (see Figure 7). A new execution meta-model must thus be created. It extends both the standard process execution meta-model and the process definition meta-model that introduces the "Deferred choice".

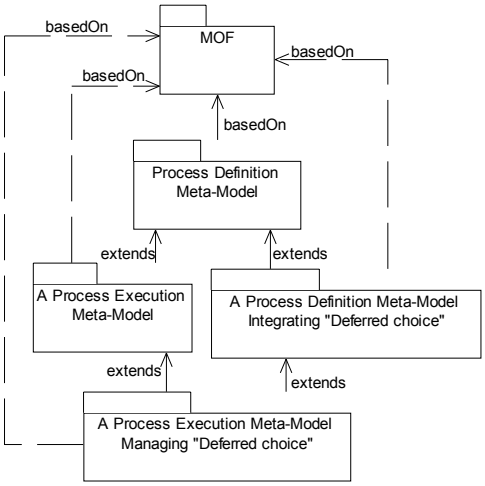

Figure 7. Extending both definition and execution meta-models

4.2. Defining the process execution meta-model

Once we have defined the organization of meta-models we may attempt to define the process execution meta-model. We place ourselves in the architecture presented in Figure 5, i.e. the execution meta-model is based on the MOF and it extends the definition meta-model.

We have chosen to reuse the concepts defined in the OMG workflow management facility in the execution meta-model. The OMG facility is a UML model while the execution meta-model is a MOF-compliant meta-model. Therefore we need to transform a UML model in a MOF meta-model. As UML and MOF formalisms are rather close this transition is easy. *UML classes* become *MOF classes*, *UML attributes MOF attributes*, *UML associations MOF associations* and *UML operations MOF operations*. The only elements that have not been moved to the meta-model level are the state machines, as the MOF does not define such mechanism (we come back to this point in the third part of this section).

The OMG specification defines the following event audits:

- A *WfCreateProcessEventAudit* records the creation of a *WfProcess*.
- A *WfStateEventAudit* records a state change of a *WfExecutionObject* (i.e. *WfProcess* and *WfActivity*).
- A *WfDataEventAudit* records a value change of a data (the concept of data is not explicit in the OMG facility).

- Finally a *WfAssignmentEventAudit* records a change in an assignment. However, for clarity reasons, we will not tackle the resources and their assignments.

These event audits correspond to the following events:

- A process instance is created (*WfCreateProcessEventAudit*) or has its state changed (*WfStateEventAudit*).
- An activity instance has its state changed (*WfStateEventAudit*).
- A data instance has its value changed (*WfDataEventAudit*).

The process, activity and data instance can therefore evolve during the execution of a process. The process instance is represented by the *WfProcess* and the activity instance by the *WfActivity*. However there is no element for representing a data in a particular process. To fill this lack we introduce the *WfData* that stores the value of a data during a particular process.

Then we have to bind the definition and execution concepts. We have therefore defined the following associations:

- An execution of a process (*WfProcess*) refers to one particular process definition (*Workflow Process Definition*), while a process definition may be executed several times.
- An execution of an activity (*WfActivity*) refers to one particular activity definition (*Workflow Process Activity*), while an activity definition may be executed several times.
- An execution data (*WfData*) refers to one particular data declaration (*Workflow Relevant Data*), while a data declaration may be referred from several process executions.

As new associations are defined, new constraints appear too. For example a *WfActivity* may request a sub-process if and only if its definition is a *SubProcess Definition*. This could be formalized in OCL in the following way:

context WfActivity **inv:**
not self.definition.oclIsTypeOf(SubProcessDefinition)
implies self.performer->isEmpty

Finally the definition of the *WfProcessMgr* has been modified. As we have seen in a former section its content is redundant with the content of the *Workflow Process Definition*. Moreover, since we have created an association between the process execution and its definition, the association between the *WfProcess* and the *WfProcessMgr* may be removed. However we have kept the *WfProcessMgr* as a process execution factory. It refers to a process definition and specify how a particular execution is created from this definition through the *create_process()* operation.

The resulting meta-model is presented in Figure 8. As it may be observed there exists a structural mapping between them. This will not be the same for every dynamic meta-model. For example a Petri nets definition meta-model defines *Place* and *Transition*, while the Petri nets execution meta-model introduces *Token* and *Marking*.

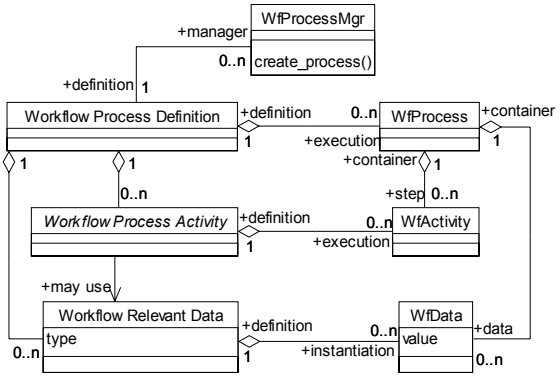

Figure 8. Binding definition and execution concepts

The integration of definition- and execution-level concerns within a same space is not a new idea. This has been already done in CPR [16] (Core Plan Representation). CPR defines plans, processes and activities. It is based on two parallel and similar trees, one that defines the design plan concept, i.e. a plan as it is expected to occur, the other allows recording execution plan, i.e. how a plan effectively occurs. 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. However the execution plan does not specify the execution rules. Definition and execution aspects may thus be bound, but the behaviour is not precisely described. The execution aspects are introduced here for statistical purposes and not to support the enactment of a plan.

A similar architecture is also proposed in the Noesis approach [4]. The authors have identified two layers in a meta-model:

- The *status-independent layer* that describes the static structure
- The *status layer* that captures the dynamic behaviour

In addition to these two layers, there is two transformation, T0 and T. T0 determines an initial status of the system starting from the status-independent layer, while T determines the next status of the system starting from the current status. In our execution meta-model the T0 transformation is performed by the *create_process()* method of *MyWfProcessMgr*, while T groups every methods that generate a change of state.

However, we do not consider, as it is done in Noesis, that the definition layer is status-independent. We have

seen that the *Workflow Process Definition* defines a status attribute. Actually the state of a process definition itself may change, but this change is asynchronous with the changes that occur in the process instances. However, as the process definition aspect is defined in a consistent manner with the execution ones, we may assume that a state change in the process definition has some impact on the process executions. For example, if a *Workflow Process Definition* comes from the *RELEASED* state to the *UNDER_REVISION* state, we may define that *WfProcessMgr* managing this definition moves to the *close* state.

Another difference is that the T0 transformation defined in Noesis implies that the link between a process definition and its execution is unidirectional, from the definition to the instances. However, nothing prevents a process definition to be modified by its instances. It could simply be that a definition records the average execution time of all of its instances. It could also be some more complex actions. For example the status of a process definition could be set to *UNDER_REVISION* when too many executions exceed the deadline. Such model could therefore be considered as an active model as "*the model reflects any modification that occurs in its subject system and it actively affects its behaviour*" [6].

4.3. Defining precise actions

In the former sections we have bound the process definition and execution meta-models. It allows integrating definition and execution aspects in a single formalism. In this way execution concepts may refer to definition ones and vice-versa. However we have not yet precisely defined how a process model is executed. We know that an instance of a process definition is created using the method *create_process()* from the *WfMgr* concept. But the side effects of this latter are not specified. We may infer from its signature that it creates and returns a *WfProcess*. But what are the values assigned to each attribute of the process instance created? Are activity instances created too? An event audit has been defined for recording such events but it is not specified when it must be created and how its fields must be filed in. Actually, what happens when a process instance is created? Similar questions may be asked for every operation defined in the meta-model. What happens if an activity instance completes? What happens if a process instance is suspended? Etc. All these questions are of the "what if?" kind. We are not interested here in the structure of the process (its definition) but in the manner a process instance may evolve from one state to another. These questions are thus only addressed to the executable concepts of the meta-model.

The answer to the question "What happens when an activity completes?" could be dependent of the template of activity. If it is a loop, the loop condition is evaluated. If it is a WHILE loop and the condition is evaluated to true or a REPEAT_UNTIL loop and the condition is evaluated to false, the activity is started once again. In any other case, the transitions following the template (i.e. the *Worflow Process Activity*) have to be evaluated to determine the next activity instances to start or resume. This algorithm may be given in such a textual form, and thus it runs the risk to be ambiguous and imprecise. Moreover it cannot be interpreted by a software component for proof-correctness, simulation or enactment purposes. A better solution is to define it in a precise manner.

The unique behavioural mechanism provided by the MOF is the operation. An operation defines a dynamic service that offers a service. The behaviour of an operation is activated through the invocation of the operation. An operation may contain parameters. At most one of them may have a return direction. An operation may be a query, i.e. the behaviour of the operation does not alter the state of the object. It may also raise some exceptions. It has a visibility (public, protected or private) and a scope (instance or classifier). In this way we may define operation signatures within MOF classes. For example we may define that the class WfProcessMgr contains the operation: *WfProcess create_process(in WfRequester requester) raises (BaseException, NotEnabled, InvalidRequester, RequesterRequired)*. This means that the operation *create_process* of *WfProcessMgr* has two parameters: the *WfRequester* requester as an input, and a WfProcess as its return value. The exception that may be raised are: *BaseException*, which allow multiple exceptions to be returned, *NotEnabled*, which indicates that the *WfProcessMgr* is disabled, *InvalidRequester*, which is raised when the requester can not be asked for this process definition, and *RequesterRequired* that is raised when no requester has been supplied.

However this is not enough for specifying precisely and exhaustively the behaviour of a process instance. First the MOF does not introduce the concept of state machines. We are therefore in the same situation that the WfMC meta-model. We can describe the state machines of processes and activities as an external artifact of the meta-model, but we cannot refer them. Second, the operations are restrited to the definition of a signature as there is no language for defining the underlying actions. Therefore we can not specify how the operation *complete()* invoked on a *WfActivity* may alter the system. There is thus a need for an action language at the MOF-level, allowing specifying the content of the operation of the meta-entities.

What could be such a mechanism? Many related efforts could be used as sources of inspiration. Some work on programming language semantics may be very

relevant. Another source of inspiration may be found in Action Semantics for the UML (AS) [1]. The AS aims at extending UML with a compatible mechanism for specifying action semantics in a software-platform-independent manner. It establishes a theory of model execution and provides a minimal set of actions for expressing behaviour (i.e. the content of a method attached to a UML class). The execution of a UML model is described by adding new entities to the standard meta-model. It may thus be compared to our execution meta-model. Some work has already been engaged to use the AS for the simulation of UML specifications [17]. Finally the sNets [8] (semantic nets), a MOF-like environment, defines the concept of semantic actions at the meta-meta-model level. In this way meta-model concepts may define their own semantic actions (in Java or Smalltalk, depending on the implementation of the environment).

5. Conclusion

In this paper we have studied how the behaviour associated to a process definition meta-model may be precisely and consistently defined. This work comes from the following observation: the WfMC Process Definition meta-model provides a formalism for describing process models but it does not integrate any execution aspect. It defines how a process may be structured, what are the different kinds of activity, the mechanisms for routing the flow of control between activities. The associated behaviour is only documented in a textual format, in order to help the designers understanding the concepts and how they will be used at execution time. The WfMC Process Definition meta-model may be compared to a programming language for which syntax has been designed but no semantics.

In order to fill this lack we have attempted to add some behavioural aspects to the WfMC meta-model. We started examining the organization of definition and execution meta-models within the MOF architecture. We have seen that the MOF may address this need. The different behaviours that may be assigned to a definition meta-model can be explicitly represented. Extensions to the definition space may also be reflected in the execution space. Meta-models, and among them process meta-models, will play a central role in the new OMG MDA organization. Standard meta-models will emerge but they will be adapted for specific needs. These extensions may impact both the structure and the behaviour. Therefore we need mechanisms for managing these two aspects in a consistent way.

Next we have defined the concepts that may be contained in an execution meta-model and their associations with the definition concepts. This execution meta-model has been inspired by the OMG Workflow Management Facility. As we have seen some of the information defined within this model were redundant with those contained within the WfMC meta-model. When binding the definition and the execution meta-models, we have observed some structural similarities. However they may also be very different. For example we have already worked on the representation of Petri nets behaviour [3]. Contrary to the example presented in this paper the definition and the execution meta-models have very different structures. The first one defines *Place* and *Transition*, while the second one introduces *Token* and *Marking*.

In the last section we have discussed a precise specification of MOF operations. They are currently restricted to the definition of a signature. There is no mechanism available for defining the actions. Extending the MOF with such mechanism would allow defining very precise process meta-models. It has been considered for long that ontologies (or meta-models) manipulate three kinds of information [2]. The terminological layer is the basic set of concepts and associations. The assertional layer is the set of axioms. The pragmatical layer contains information that does not fit in the two former layers. The terminological layer is defined using the MOF meta-meta-model and the assertional layer using OCL. At present the pragmatical layer itself is going to be organized and some part are made explicit. XMI provides a mechanism for serializing meta-models and models. A RFP has been submitted for defining a meta-model of UML diagrams [13]. The specification of an action language at the MOF-level would be a further step towards the removal of the pragmatic layer.

In this paper we have attempted to make explicit the behaviour of a process meta-model in a consistent manner with its definition. Making the behaviour explicit would allow manipulating it, i.e. modifying it for specific needs. Moreover a precise specification of the behaviour at the meta-model level could be used to provide generic tools for simulation, enactment or proof-correctness. The MOF architecture provides an integration framework for all meta-models in the software development scene. This integration was so far restricted to the separation and combination of separate structural aspects. As the variety of processes grows (BtoB, EAI, workflow, business process, etc.) and their importance in the information system increase, the integration of behaviour becomes a major need.

Defining in a consistent manner both the definition and execution meta-models may also lead to many benefits in process management. In the WfMC reference model the design of a workflow and its execution are separate activities. Weaving these two aspects may allow round-trips between them. We may immediately find some practical applications to such a capability. Changes in the definition may impact the under way executions. This could even allow the process definition to be built as the

executions go along. Executions may also alter the process definition.

6. References

[1] Alcaltel, I-Logix, Kennedy-Carter, Kabira Technologies, Inc., Project Technology, Inc., Rational Software Corporation, Telelogic AB, "Action Semantics for the UML", OMG Document ad/2001-03-01, March 2001, http://cgi.omg.org/cgi-bin/doc?ad/01-03-01.

[2] Bézivin J., "Who's Afraid of Ontologies?", OOPSLA'98 Workshop: Model Engineering, Methods and Tools Integration with CDIF, Vancouver, October 1998, http://www.metamodel.com/oopsla98-cdif-workshop/bezivin1/.

[3] Breton E., Bézivin J., "Towards an Understanding of Model Executability", Proceedings of the Second International Conference on Formal Ontology in Information Systems, Ogunquit, Maine, USA, October 2001.

[4] Dominguez E., Rubio A., Zapata M., "Meta-modelling of Dynamic Aspects: The Noesis Approach", ECOOP'2000 Workshop: International Workshop on Model Engineering, Nice, June 2000, http://www.metamodel.com/IWME00/articles/rubio.pdf.

[5] Giese H., "Towards a Dynamic Model for the UML", 14th Annual ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages, and Applications, Workshop: Rigorous Modeling and Analysis with the UML: Challenges and Limitations, Nov. 1999, http://www.upb.de/cs/hg/archive/1999/OOPSLA99/oopsla99.html.

[6] Greenwood R.M., Robertson I., Snowdon R.A. & Warboys B.C., "Active Models in Business", Proceedings Business IT Conference, Manchester, 1995, ftp://ftp.cs.man.ac.uk/pub/IPG/grsw95.ps.

[7] Hollingsworth D., "Workflow Management Coalition The Workflow Reference Model", Document WfMC-TC-1003, November 1994, http://www.wfmc.org/standards/docs/tc003v11.pdf.

[8] Lemesle R. "Techniques de modélisation et de Méta-Modélisation" PhD Thesis of the University of Nantes, October 2000, http://www.sciences.univ-nantes.fr/info/lrsg/Recherche/mda/richard.pdf.

[9] OMG, "Common Warehouse Metamodel (CWM) Specification", OMG Document ad/2001-02-01, February 2001, http://www.omg.org/technology/cwm/index.htm.

[10] OMG, "Meta Object Facility (MOF) Specification v1.3", OMG Document formal/2000-04-03, March 2000, http://www.omg.org/technology/documents/formal/mof.htm.

[11] OMG, "OMG Unified Modeling Language Specification v1.3", OMG Document formal/2000-03-01, March 2000, http://www.omg.org/technology/documents/formal/uml.htm.

[12] OMG, "UML Extensions for Workflow Process Definition Request For Proposal", OMG Document bom/2000-12-11, December 2000, http://cgi.omg.org/cgi-bin/doc?bom/00-12-11.

[13] OMG, "UML 2.0 Diagram Interchange RFP", OMG Document ad/2001-02-39, March 2001, http://cgi.omg.org/cgi-bin/doc?ad/01-02-39.

[14] OMG, "Workflow Management Facility Specification, v1.2", OMG Document formal/2000-05-02, April 2000, http://www.omg.org/cgi-bin/doc?formal/2000-05-02.

[15] OMG, "XML Metadata Interchange (XMI) Specification v1.1", OMG Document formal/2000-11-02, November 2000, http://www.omg.org/cgi-bin/doc?formal/2000-11-02.

[16] Pease A., "Core Plan Representation, version 4", November 1998, http://reliant.teknowledge.com/CPR2.

[17] Pennaneac'h F., Sunyé G., "Towards an execution engine for the UML", UML 2000 Workshop: Dynamic Behaviour in UML Models: Semantic Questions, York, October 2000, http://www.disi.unige.it/person/ReggioG/UMLWORKSHOP/Pennaneach.pdf.

[18] Soley R. and the OMG Staff Strategy Group, "Model Driven Architecture", November 2000, ftp://ftp.omg.org/pub/docs/omg/00-11-05.pdf.

[19] Van der Aalst W.M.P., Barros A.P., ter Hofstede A.H.M., Kiepuszewski B., "Advanced Workflow Patterns", 7th International Conference on Cooperative Information Systems (CoopIS 2000), volume 1901 of Lecture Notes in Computer Science, pp 18-29. Springer-Verlag, Berlin, 2000, http://tmitwww.tm.tue.nl/staff/wvdaalst/Publications/p105.pdf.

[20] WfMC, "Workflow Management Coalition Interface 1: Process Definition Interchange Process Model", Document WfMC-TC-1016-P, October 1999, http://www.wfmc.org/standards/docs/TC-1016-P_v11_IF1_Process_definition_Interchange.pdf.

[21] WfMC, "Workflow Management Coalition Terminology and Glossary", Document WfMC-TC-1011, February 1999, http://www.wfmc.org/standards/docs/TC-1011_term_glossary_v3.pdf.