AMFIBIA: A Meta-Model for the Integration of Business Process Modelling Aspects*

Björn Axenath, Ekkart Kindler, Vladimir Rubin

Software Engineering Group, University of Paderborn,
Warburger Str. 100, D-33098 Paderborn, Germany
[axenath|kindler|vroubine]@uni-paderborn.de

Abstract. *AMFIBIA* is a meta-model that formalizes the essential aspects and concepts of business process modelling. Though AMFIBIA is not the first approach to formalizing the aspects and concepts of business process modelling, it is more ambitious in the following respects: First, it is independent from particular modelling formalisms of business processes and it is designed in such a way that any formalisms for modelling some aspect of a business process can be plugged into AMFIBIA. Therefore, AMFIBIA is *formalism-independent*. Second it is not biased toward any aspect of business process and the different aspects can be, basically, considered and modelled independently of each other. Moreover, it is not restricted to a fixed set of aspects; further aspects of business processes can be easily integrated. Third, AMFIBIA does not only name and relate the concepts of business process modelling, as it is typically done in ontologies or architectures for business process modelling. Rather, AMFIBIA also captures the interaction among the different aspects and concepts, and therefore fully defines the dynamic behaviour of a business process model, with its different aspects modelled in different notations. To prove this claim, we implemented a prototype of a formalism-independent workflow engine based on AMFIBIA: This workflow engine, also called AMFIBIA, is open for new aspects of business process modelling and new modelling formalisms can be added to it.
In this paper, we will present AMFIBIA and the prototype workflow engine based on this meta-model and discuss the principles and concepts of its design.

1 Introduction

Today, there is a good understanding on what business processes are, how they should be modelled, and which aspects and concepts are important in such models. In the literature on business process modelling and workflow manangement of the last ten years, the three aspects of *control*, *information*, and *organization* have consistently been identified as the most important aspects of a business

* This is a revised version of the report "The Aspects of Business Processes: An Open and Formalism-Independent Ontology" released in April 2005.

Dagstuhl Seminar Proceedings 06291
The Role of Business Processes in Service Oriented Architectures
http://drops.dagstuhl.de/opus/volltexte/2006/823

process model. Sometimes called perspectives or views and sometimes with different names, these three aspects have been identified quite early when the topic of business process modelling and workflow management became an issue. The resulting insights were formulated as architectures, ontologies or meta-models for business process modelling [35, 20, 24].

Though there is a wide agreement on theses aspects and concepts, the concrete formalisms, notations, and, in particular, the business process modelling tools vary and are not compatible to each other. The reason is that each formalism and notation set out from one or two particular aspects and later on integrated the concepts for other aspects, or they were focussed on solving one particular problem and later on were generalised. Moreover, the formalisms for the different aspects are tightly integrated with each other, though the aspects are conceptually independent of each other. Therefore, the fact that the different aspect are, basically, independent of each other is concealed by these approaches. The actual aspects, the concepts modelled in these aspects, as well as their independence get blurred and distorted by the concrete formalisms and notations of the tools. And most formalisms and tools are biased towards one aspect and neglect others. In particular, integrating new aspects or new formalisms in existing tools and exchanging models among different tools is virtually impossible.

Altogether, the architectures, frameworks, ontologies, or meta-models for business process modelling proposed so far provide a good overview on which aspects and concepts need to be modelled and to compare different formalisms and tools on a high conceptual level. These models, however, do not deal with the details of the aspects, their concepts, and, in particular, the interplay of these aspects independently from a particular formalism. A formalization of this finer fabric of the aspects and concepts of business process modelling is still missing. Actually, this insight struck us at a workshop on 'XML Interchange Formats for Business Process Models' [31]: At this workshop, different proposals for exchange formats for business process models were made. Some where based on a particular notation for business process models such as BPEL, EPCs, or Petri nets; others set out from XML technologies such as XMI or GXL. Altogether, there was a wide agreement that it would be nice to have a standard interchange format for business process models; but, the variety of different proposals showed that there is a long way to go before such a general standard will be generally accepted. In fact, the essential problems are not so much in the syntactic differences or in the different XML technologies underlying the proposals. The real problems are the different comprehension of what business processes are, of what the important aspects of business processes are, and of which formalisms should be used for representing them. Most proposals for interchange formats are biased towards one or two aspects of business process models and are neglecting others; and they use – or at least favour – particular formalisms.

This was the starting point of our research on *AMFIBIA*: *A Meta-Model for the Integration of Business Process Modelling Aspects*. AMFIBIA should not only capture the rough outline of the different aspects and concepts of business process models, but also the fine details and the exact interplay of the different

concepts – independently from a particular formalism. In order to be formalism-independent, there needs to be an interface for mapping a formalisms to the concepts of a particular aspect. Moreover, AMFIBIA should not be biased toward one aspect of business process modelling and should be open for integrating new aspects.

In this paper, we present AMFIBIA and formalize it in terms of a UML meta-model; for proving that these concepts work, we[1] have implemented a workflow engine based on the concepts and the meta-models of AMFIBIA, which is also called AMFIBIA: a formalism independent workflow engine that is open for adding new aspects and new formalisms. Altogether, the objectives of AMFIBIA are the following:

- It should cover all basic aspects of business process models and should not be biased toward or focused on one of these aspects.
- It should be open so that other aspects can be easily added and integrated to it.
- In particular, there should be clear interfaces for the different aspects and a mechanism for their integration.
- It should be independent of a particular formalism or notation for business process models. But, it should be easily possible to map existing business process modelling notations to it.

The first ideas of AMFIBIA were already presented in [5]. At that time, the focus was on structuring and relating the concepts; the interaction between the different aspects were quite informal at that time. Here, we present the structural meta-models of AMFIBIA as well as the interfaces and the interaction between the different models. This – together with the prototype implementation of a workflow management system – proves that the concepts of AMFIBIA work. In order to model the dynamics for each aspect, our meta-model provides *automata* defining the behaviour for each aspect, where the state changes of this automata are triggered by *events*. The events may be shared by automata for different aspects. The overall behaviour of all aspects is defined by executing all these automata concurrently, where shared events are synchronized.

Actually, the AMFIBIA project has another quite different objective: Due the independent aspects, AMFIBIA is a good example for *aspect-oriented modelling* (AoM). Though there are many ideas and approaches toward aspect-oriented modelling, there is no final and mature technology for aspect oriented modelling in the context of the model driven architecture (MDA) yet. We think that AMFIBIA is a good example for studying the concepts necessary in the context of MDA and to develop and refine this technology from a software engineering point of view. This, however, is not the focus of this paper[2].

[1] Actually, it was a group of eight students who have implemented AMFIBIA in a one-year master's project (see Acknowledgement in the end of this paper).

[2] Note that, in the literature on aspect oriented modelling, our aspects would often be called 'concerns' rather than aspects.

2 Concepts of Business Process Modelling

In this section, we introduce the basic concepts and terminology used in business process modelling and define a controlled vocabulary for the domain of business process modelling. Later in this paper, we will formalize these concepts as a UML meta-model. In the literature, there are many different proposals using different terms, which are not consistent and no terminology is fully accepted. Our definitions and terminology has been strongly influenced by the work of the Workflow Management Coalition (WfMC) [20, 22], by van der Aalst and van Hee [37], and by Leymann and Roller [29].

2.1 Overview

A business process involves a set of *activities* that are executed in some enterprise or administration according to some rules in order to achieve certain goals. A *business process model* is a more or less formal and more or less detailed description of the persons and artefacts involved in the execution of a particular business process and its activities as well as of the rules governing their execution. An *instance*, i. e. a particular execution of a business process model, is often also called a *business process*. In order to avoid confusing instances and models, we call an instance of a business process model a *case*, and we will, henceforth, use the term business process informally only.

As for models in general, business process models are used for different purposes:

- Documentation of the business process.
- Better understanding of the business process.
- Collaborative design of the business process.
- Communication and teaching of the business process.
- Analysis and verification of the business process.
- Optimisation and re-engineering of the business process.
- Computer support and automated execution of the business process (which is often called workflow management or enactment).

The purpose of a model governs the choice of the formalism and notation as well as the level of abstraction or detail for modelling a business process.

It is now well-accepted that there are three basic aspects of business processes that can be modelled and investigated more or less independently of each other. These aspects are shown in Fig. 1. The *control aspect* basically describes the order in which the different activities are executed. The *organization aspect* describes the organization structure and, in particular, the *resources* and *agents*, and in which way they are involved in the business process. The *information aspect* describes the information that is involved in a business process, how it is represented, and how it is propagated among different activities. The details of theses aspects, the concepts modelled in each particular aspect, and how the aspects are integrated will be discussed below.

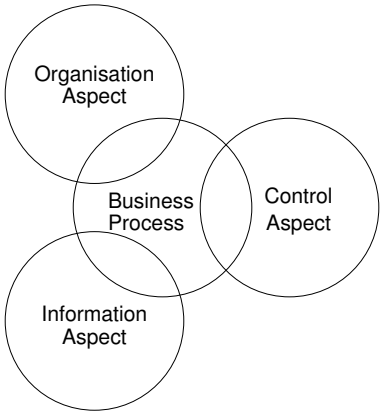

Fig. 1. A business process and its basic aspects

In the literature on business process modelling, the most salient aspect of a business process is the control aspect. In many modelling formalisms and notations, this aspect is used for integrating all other aspects. Actually, in ARIS it was introduced for integrating all other aspects [36]. In our proposal, we do not use the control aspect for the integration of the other aspects. Rather, we single out the concepts that are common to all aspects. We call this the *core* of a business process, which are basically the *activities* of the process. An activity itself is an instance of some particular *task*; only when a particular case is executed the tasks will be instantiated to an activity.

A *task* comprises pieces of work that conceptually belong to each other. A task can be either *atomic* or *compound*. An atomic task is not split into further parts on the given level of abstraction. Dependent on the purpose of the model, an atomic task can be associated with a procedure or an program that supports the execution of this task; this, however, is subject to a different aspect already: the *application aspect*. A compound task refers to a *subprocess*, which defines the details of this task; this can be defined by another business process. Actually, the distinction of atomic and compound tasks belongs to the *structuring aspect* of a business process, which will not be discussed in detail here.

Altogether, the concepts *activity*, *task*, *process* (short for business process), and *case* belong to the *core* of business process modelling, which occur in virtually all other aspects.

In the following sections, we will discuss the main aspects of business process modelling, where we concentrate on the aspects necessary for enacting business processes. Note that we omit the very important *functional aspect*, which defines the objectives and goals that are achieved by a business process or a task. Since, this aspect is not needed for enacting business process, we do not formalize it here.

2.2 Control aspect

The control aspect defines the order in which the tasks of a business process are instantiated and in which the corresponding activities are executed. Note that the order of the execution does not need to be sequential; it can be a partial order representing the dependencies among the activities, some of which may be concurrent.

For defining this order, the formalism refers to the tasks defined in the core of the business process. There are many different ways, formalisms, and notations for defining the order in which tasks, resp. the corresponding activities must be executed. In Sect. 3.1, for example, we will use Petri nets for modelling the control aspect. In order to be universal, however, we do not fix a particular formalism for defining the control aspect of a business process. We assume only that there is a concept of a *state*. In a given state, it must be clear which tasks are *activated* or enabled (i. e. which tasks can be instantiated to activities), how the *instantiation* of a task to an activity changes the state, and how the *termination* of an activity changes the state. We will discuss this in more detail later in Sect. 3.3.

2.3 Information aspect

The information aspect of a business process model defines the information involved in a business process as well as the propagation of information among different activities. All information involved in a business process can be considered to be *documents*[3], where a document is an artefact representing some piece of information. The information aspect of a business process basically defines the structure of the involved documents and their relation. Moreover, the information aspect defines how documents are propagated among activities.

Similarly to processes and cases and to tasks and activities, we must distinguish between *document instances* (documents for short) and *document models*, where a document model defines the structure of a document instance. The *information model* comprises all document models as well as the relation among the different documents. As for tasks, a document can be *atomic* or *compound*. An atomic document is an unstructured text or piece of data (resp. the structure is not represented in the model), whereas a *compound document* is structured and consists of two or more *sub-documents*.

2.4 Organization aspect

The organization aspect of a business process model defines the structure of the organization in which the business process is executed, its *organization units* and the *relations* among them; and it defines the *resources* and *agents* within these

[3] In some systems, information is stored in a relational database. In that cases, we can consider the tuples in the database as documents. This point of view allows us to unify the terminology.

organization units, where we use the term agents in order to refer to persons as human resources. Moreover, the organization aspect of a business process model defines which resources and agents could possibly execute a particular task resp. activity; these are called the possible *assignments* for that task. In a business process model each task will be equipped with a *resource descriptor*, which defines the possible assignments once the task becomes enabled.

The structure of the organization is modelled in the *organization model*. Note that the organization model captures only that part of the organization which is more or less fixed, such as departments and groups. It does not deal with the concrete agents and resources, which change much more rapidly. Therefore, the possible assignments of agents and resources to some task are defined by a *resource descriptor*, possibly via their *positions*, *roles* or via *relations* (such as substitute) among resources.

When it comes to the execution of a business process model in a workflow management system, the concrete resources and agents, their positions and roles will be maintained by some administrator, such that the workflow management system can assign tasks to the concrete agents.

2.5 Further Aspects

The three aspects mentioned above are generally agreed to be the most important aspects of business process. But, depending on the context, there can be more or even less aspects. Which of them are applicable or relevant depends on the context and the purpose of the model. Figure 2 shows some additional aspects that will be briefly discussed below. There could be even more aspects, such as

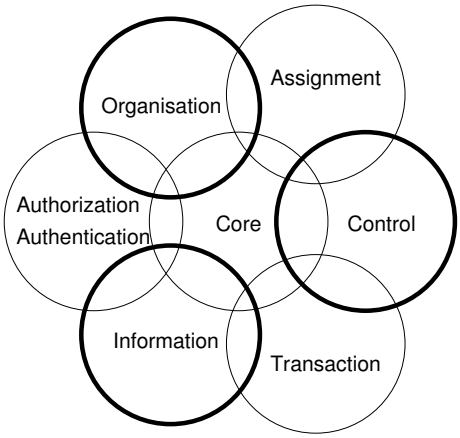

Fig. 2. More aspects of business processes

the *structuring aspect*. But, we do not deal with the details of this aspect here.

The *assignment aspect* defines in which way tasks are assigned to resources or agents. Note that the organization aspect does not define who will do some work, it defines only who could do some work. The assignment policy defines how the work will be assigned to the resources. The choice could be either with the agents who choose it from some work-list, or the choice could be made by the workflow management system. The first policy is called *pull policy*, the second is called the *push policy*. And there are all kinds of assignment policies in between, which could be defined by a sophisticated assignment scheme, in order to define *escalation policies*. The assignment aspect of a business process captures the concepts for defining assignment policies for tasks.

The *security aspect* defines which agents or resources may read, access, and change information. Moreover, it makes sure that documents are authentic.

The *transaction aspect* defines which chunks of work and associated changes on documents must appear to be executed in isolation from the perspective of other business processes. This involves transaction protocols, compensation schemes, or nested transactions known from classical transaction theory [8, 18].

In this paper, we will not go into the details of these special aspects. But, we will demonstrate that our meta-model allows us to add new aspects.

2.6 Models and Instances

In the presentation of the concepts of business processes, we have dealt with two kinds of concepts. The first kind were the concepts occurring in the business process model such as the process itself, its tasks, the document models, the roles etc. We call these *modelling concepts*. The second kind are the instances of the first ones such as cases, activities, document (instances), and resources. We call these concepts *instance concepts*.

In the business process models, there will be only modelling concepts. The instance concepts are necessary only for the definition of the dynamic behaviour and for discussing the dynamic behaviour of the modelling concepts – and when implementing a workflow management system. In the following discussions and, in particular, in the meta-model, we will carefully distinguish between concepts of these categories, which are often confused: when someone is talking about a business process, one can never be sure whether he is talking on the model or on the case. Still the distinction is very important.

3 Structural Integration of Aspects and Formalisms

Up to now, we have presented the concepts of the business process modelling aspects. Now we formalize these concepts in terms of UML models and show how the aspects can be integrated with each other and how formalisms can be mapped to an aspect. These techniques are formalized in the following subsections with the help of an example. The dynamic interaction of the aspects is defined separately in the next section as it is the major contribution of our approach.

3.1 A Conference Trip Example

Before we explain how the concepts from Section 2 are formalized, we introduce an example to make our explanations better understandable. This example describes which administrative tasks have to be done by a member of a company to perform a business trip. In our example, we selected the control and the information aspect to show how business process can be described by different aspects. Furthermore, the order in which the aspects are presented is arbitrary. Nevertheless, we start with the control aspect as is the more common aspect.

The Control Aspect of the Example In Fig. 3, we can see the control aspect of the business process model. The formalism is a special version of Petri nets called *workflow nets* [37]. It defines the different tasks of a conference trip and the order in which they are executed. The tasks are the transitions (represented as rectangles) of the Petri net. A Petri net defines this control by a firing rule referring to a *marking*, which is a number of tokens on its places (represented as black dots in the circles). Initially, there is only one token on the initial place *in*. At this stage, only transition "apply for trip" is enabled. After starting and finishing the corresponding task, the transitions "support trip", which corresponds to the task of a superior countersigning the trip application, and the transition "book trip" are enabled. This means that these two tasks can be executed concurrently. Note that the actual trip may be made only if the trip application was approved before. After the trip, the employee may apply for reimbursement of the travel expenses. Note that, at this stage, there might be an iteration: If the bills for the trip are rejected, the billing activity will be repeated. The process is finished, when a token arrives at the place *out*.

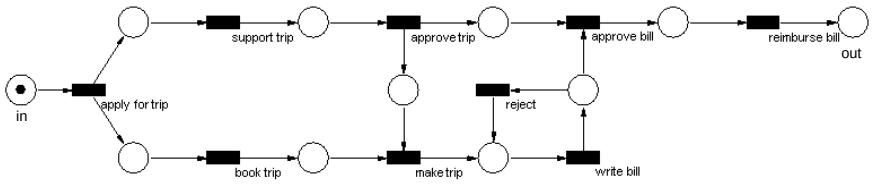

Fig. 3. Control aspect of the example

The Information Aspect of the Example In our example, the information needed for the business trip is stored in a database and is modelled by an ER diagram as shown in Fig. 4.

For the process, it is necessary to compose documents out of this data. To apply for the trip, the user has to fill the table *Trip Application* out with the

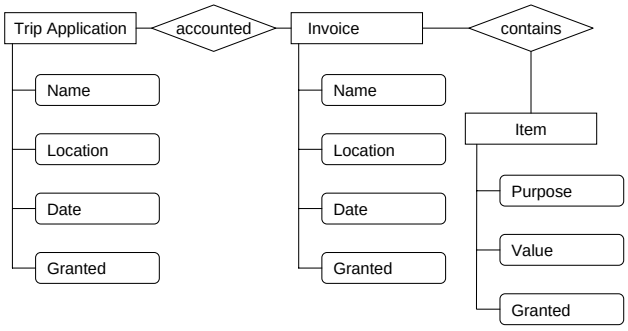

Fig. 4. The Information Aspect of the Example

attributes *Name*, *Location* and *Date*. When the trip is approved, the attribute *Granted* is filled out. After the trip, the bill is written by creating an entry in the *Invoice* table, which has to be linked to the corresponding entry in the *Trip Application*. Furthermore, to write the bill, all expenses have to be listed in the *Item* table. All these entries have to be linked to the *Invoice*. When the Bill is approved, it can be reimbursed.

Notice, that we write about tasks that are also mentioned in the control aspect of our example. These tasks are not defined in the control aspect but in the core of the business process. The control aspect and the information aspect refer to tasks which are defined in the core, as the tasks are relevant for all aspects.

3.2 The Core of Business Process Modelling

In Section 2.1 we motivated already that we use a core to integrate the aspects. In this subsection we formalize the core.

The core consists of concepts that are common to all aspects (see Figure 5). There is, first of all, the *business process model* (BPM) itself which consists of a set of *tasks*. A *case* is an instance of a particular business process. While running a case, different tasks will be instantiated to *activities*; each activity corresponds to exactly one task. As we have motivated in Section 2.6, we distinguish between models and instances. We do this by introducing an *instance-of* relation. Furthermore we group the elements by using packages. The left package shows the concepts of the model itself, whereas the right package shows the concepts concerning the instantiation of these models when the models are executed.

Every aspect adds new concerns to the elements of the core. We introduce the UML stereotypes *aspect* and use packages to show that an element is extended by an aspect, like it is shown in Figure 6. Here, the *BPM* from the core is extended by the control aspect, which is denoted by the package name. The meaning of the stereotype *aspect* is, that there is a relation to an element in the core as it is shown in Figure 7. Consequently, the *aspect-of* relation defines that the element

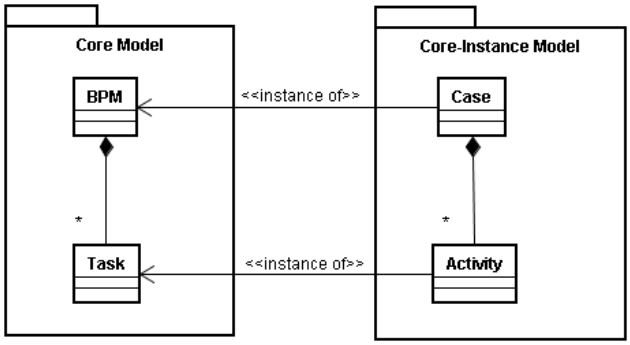

Fig. 5. The Core of the AMFIBIA-Model

from the core has a corresponding element in the aspect. Here we can also see, that the *BPM* from core consists of several other classes defined in the aspects.

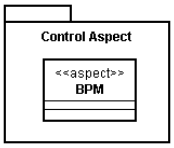

Fig. 6. BPM of Core in Control Aspect

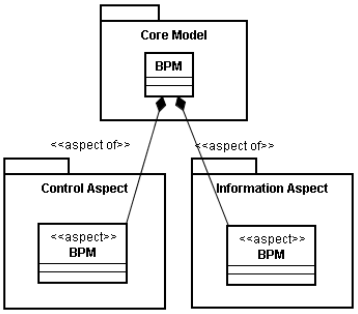

Fig. 7. Aspect-of relation

3.3 Aspects and Formalisms

The integration of the aspects shown in the previous subsection is still formalism-independent. In the next paragraphs we will formalize the aspect independent meta-models and show how the formalism-dependent meta-models can be integrated, like it is illustrated in Figure 8. We will exclude the organization aspect from this paper as all ideas can be explained by the help of the information aspect and the control aspect. Furthermore, the concepts of integration of the organization aspect is similar to integration from the information aspect. Like in the example, we start with the control aspect.

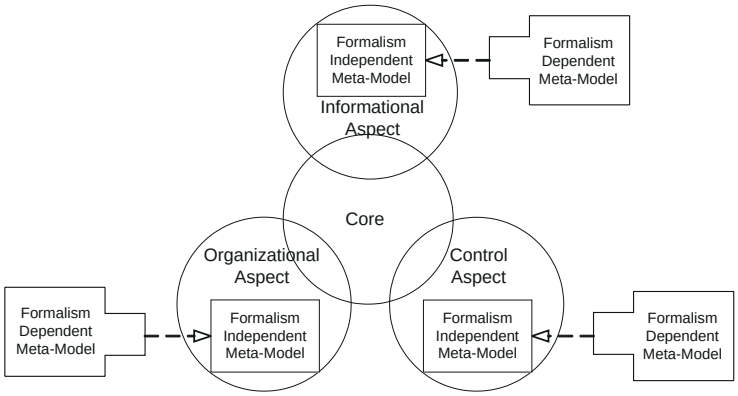

Fig. 8. Mapping of the formalism-dependent meta-models

Control Aspect In this section, we present the meta-model for the control aspect of business process models. To this end, we refer to the concepts of the core (see Fig. 5) and extend them by additional features. These are shown in the UML diagram in Fig. 9.

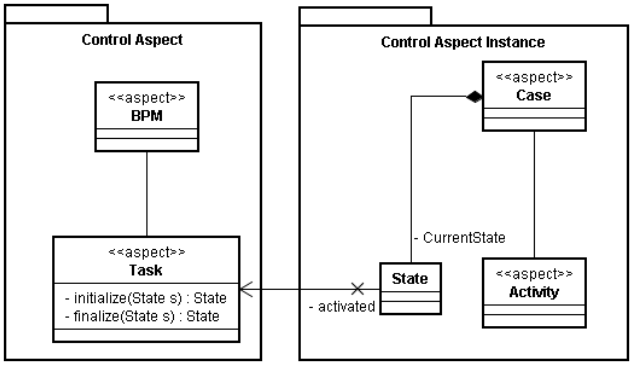

Fig. 9. The meta-model for the control aspect

For the control aspect, a process model is equipped with *initial* and *final tasks*. The initial tasks define those tasks that initiate a new case. The final tasks identify those tasks that terminate the execution of the corresponding case.

In order to define the process control, there is the concept of a state, which actually sits in the middle between the model and the instance concepts of business processes. Each case has a current state, which in turn defines the tasks

that could possibly be started in the current state; these are called the *activated* tasks. Each task defines, how the initialization of this tasks changes the state and how the finalization of the corresponding activity changes the state of the case. Moreover, a case consists of a set of activities that are active at a particular moment, and it consists of activities that are finished already.

Thus far, the meta-model for the control aspect is independent of a particular formalism for modelling the control. It only requires that there is a concept of state and state changes. This can be implemented by different formalisms. Here, we show how Petri nets can be used for implementing the control aspect. Figure 10 shows the meta-model of Petri nets implementing the concepts of the control aspect of business processes. It consists of transitions and places. A marking consists of a multi set of places. The transitions implement tasks, the markings implement states, and the firing rule of Petri nets implements the state changes. The method initialize removes one token from each of its input places, and the method finalize adds a token on each of its output places. The enabledness of a transition in a particular marking implements the set of activated tasks.

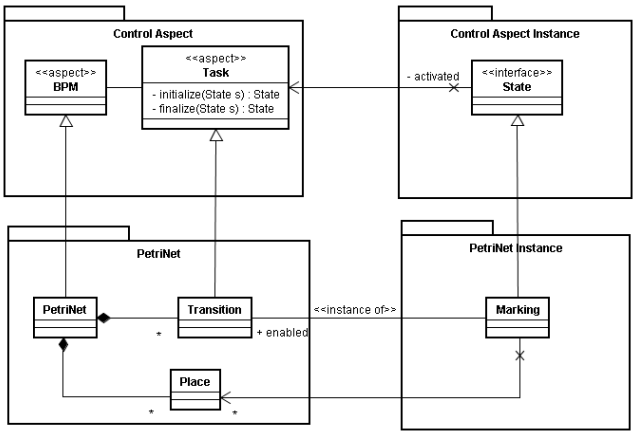

Fig. 10. A Petri net implementation

Here, we used Petri nets for implementing a formalism for the control aspect of business process models. But, it is easy to see that any other formalism that has a semantics based on transition systems can be used for implementing the control aspect.

Information Aspect The information aspect (see Fig. 11) describes which documents are needed to instantiate a *Task* to an *Activity*. Therefore a *DocumentDescriptor* is introduced, which describes what kind of document is needed for the execution of a task. The type of document is given by the interface

DocumentType. Furthermore, for the selection of a *Document* the *DocumentDescriptor* regards further *Constraints* according to the context, which is given by the *Case*.

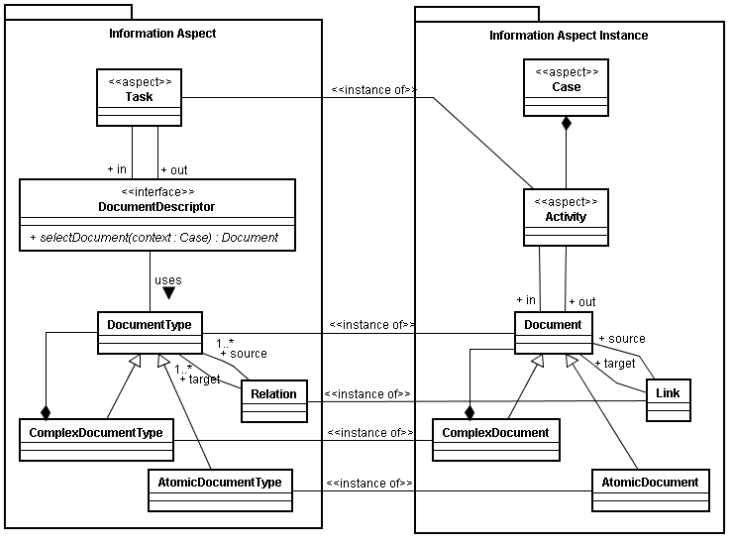

Fig. 11. Information aspect

Documents are usually structured so that we differentiate between *AtomicDocument* and *ComposedDocument*. The meaning of atomic is, that in the context of the business process the document cannot be split into parts. The *ComposedDocument* implements its composing property by an association to its parent class *Document*. Documents are related to other documents by *Links*. Both, *Documents* and *Links*, have definitions called *DocumentTypes* resp. *Relations*. Analogously to *Documents*, *DocumentTypes* can be either atomic or complex. An example for a *Relation* is a dependency. According to our drawn distinction of model and instance, *Documents* and *Links* belong to the instance part and *DocumentTypes* and *Relations* belong to the model part.

Figure 12 shows an excerpt of a meta-model for the entity relationship model, which was used for our example. It also shows how this formalism-specific model is integrated in the information aspect. Notice, the documents being used for our conference trip are tuples contained by our model. Consequently, for the mapping from the formalism-independent model to the formalism-dependent model we need a construct extracting the documents from the data described by the model. In the formalism-dependent model, this is done by SQL queries, which create result sets. Consequently, result sets are documents. Depending on the inner structure of the query, the documents are atomic or composed.

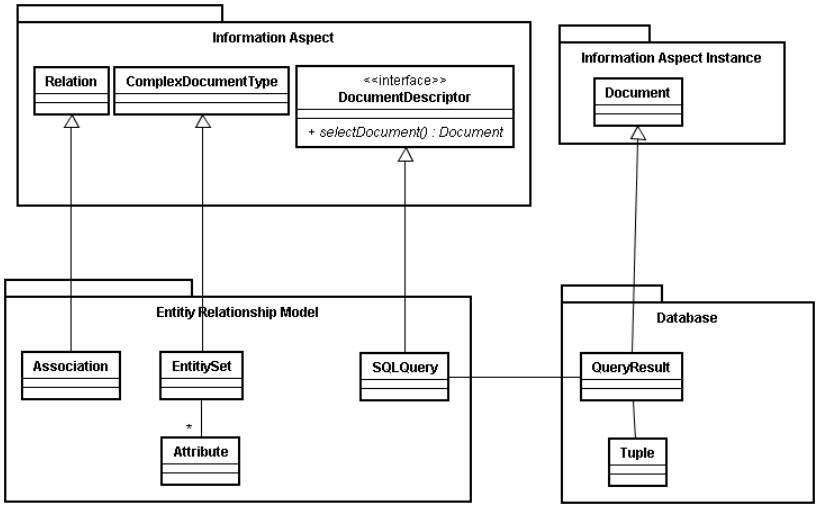

Fig. 12. Excerpt of Meta-Model for Entity-Relationship-Model with the Mapping to Information Aspect

4 Dynamic Interaction of Aspects

In the previous sections, we presented the structural meta-models for the core part of business process models and the meta-models for two different aspects of business process models. We have shown how the aspects can be conceptually integrated and how the meta-models of formalisms can be mapped to the meta-models of aspects.

In this section, we deal with the concepts of the *behaviour of aspects* and the concepts of the *dynamic integration of aspects*. Moreover, we present the behavioural models and discuss how to implement our ideas in a *workflow engine*. This workflow engine has to support integration and dynamic interaction of business process modelling aspects. The prototype of the workflow engine was implemented in a Project Group AMFIBIA at the University of Paderborn.

The main points discussed in this section are the execution of processes and the *dynamic interaction* of aspects. A business process model, which includes different aspects has to be instantiated and executed, i.e. a *case* is created. During the case execution, it is decided, which *tasks* are enabled: all the aspects are asked which tasks are enabled from their point of view. This decision is specific for each aspect, it depends on the available information, resources, control flow, etc. The tasks enabled for all aspects are instantiated, thus, the *activities* are started. The aspects fill the activities with information, resources, or decide who will execute them (this functions are also aspect-specific). Thus, the main challenge here is to come from the conceptual integration of aspects in meta-models and models to the concepts of dynamic integration of aspects and, furthermore, practical ideas about execution of them in a working workflow system.

4.1 Aspects Automata and Synchronization

In the following, we discuss the functionality described above in more details and with the help of the examples. Here, we present a set of simplified automata models. These automata have some restrictions: e.g. they do not support the concurrent executions of activities of a single case. But they clearly show the idea of the modelling of the dynamic behaviour and the integration of the behaviour of different aspects.

The basic functionality of the workflow engine is covered by the *core*, it is the essential component. The core is responsible for executing cases and activities. The new aspects are plugged into the core, their execution is synchronized with the execution of the core.

For describing the behaviour of the core, we identify the *events* that can be emitted. First, we start with the events produced during the lifecycle of a case. So, these events occur for every case and they are: "start Case", "finish Case", "request Tasks", "receive Tasks", "create Activity", "finish Activity". The events define the points of a case execution, which are especially interesting in respect to aspect coordination or to the functionality of the core[4].

The case execution behaviour in the core can be modelled as a *case core automaton*, see Fig. 13. When a case is started, the automaton goes to the state *initial*; then, the core requests enabled tasks from aspects and goes to the next state. After the enabled tasks are received, i.e. event *receive Tasks* is emitted, the core creates an activity. When the activity is started, the core goes to the next state and waits for the *finish Activity* event. Then, a set of enabled tasks is requested again and the whole loop of receiving tasks and executing activities is repeated until the case is finished.

Like we model the case execution behaviour in the core, we model the case execution behaviour in the aspects. The automaton of the control aspect, called *case control automaton*, is shown in Fig. 14. It has such events as "request Tasks", "calculate Tasks" and "receive Tasks".

After we have presented the core and the control aspect automata, we can explain the ideas of the *synchronization of aspects* on these examples. As it was mentioned earlier, aspects are synchronized with the core and with each other. First of all, we assume that all the automata are executed in the system *concurrently* and that they are started synchronously, when the case is started. The events in the core will be synchronized with the events of the aspects. In our examples, the events that are synchronized have identical names and we use to add "*(sync)*" to the end of the names. For example, event "request Tasks" in the core has to be synchronized with the event "request Tasks" in the control aspect; the same is about "receive Tasks" and other events.

The synchronization works the following way: if the core can execute an event, for example "request Tasks", it waits until all the aspects that have the same event can execute it. Then, the core and the aspects execute this event

[4] We can draw here the analogy to the formal approaches from the area of aspect-oriented programming (AOP) [13], where events are abstractions of the *points of interest*.

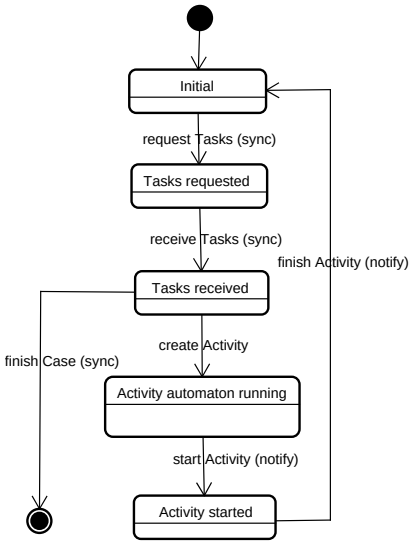

Fig. 13. Case Core Automaton

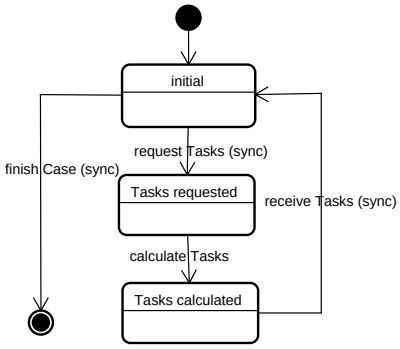

Fig. 14. Case Control Automaton

concurrently and go to the next state. After it, the core can execute the next event "receive Tasks" and starts waiting. When the "request Tasks" event is executed by the aspects, the aspects do their job; for example, the control aspect calculates the enabled Tasks ("calculate Tasks" event) depending on the semantics of the control aspect formalism and on the process model. Then the aspects execute their further events; as soon as all the aspects execute the event the core is waiting for, the core and the aspects go to the next state again.

In our examples in Fig. 13 and in Fig. 14, the synchronization is done for the following events: "request Tasks", "receive Tasks", and "finish Case". The automata of the *information aspect* looks exactly like the automata of the control aspect, the synchronization is based on the same events, but the calculation of tasks expressed in the corresponding event is based on the information model.

Along with the execution of cases, the engine manages the execution of activities. Activities are created within a case. The activity execution behaviour in the core is also modelled as an automaton shown in Fig. 15. This automaton contains two events: "start Activity" and "finish Activity".

When the activity is started, the case has to be notified about it and has to start waiting for this activity to be finished. Thus, there is another type of synchronization between the automata: it synchronizes the events of the activity core automaton with the events of the case core automaton. We use to add "*(notify)*" to the end of the names of such events. For example, events "start Activity" and "finish Activity" in the activity core automaton synchronize with the corresponding events in the case core automaton.

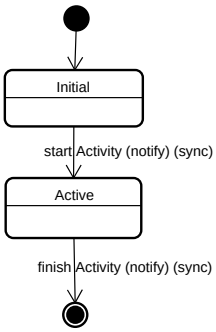

Fig. 15. Activity Core Automaton

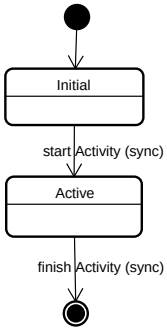

Fig. 16. Activity Control Automaton

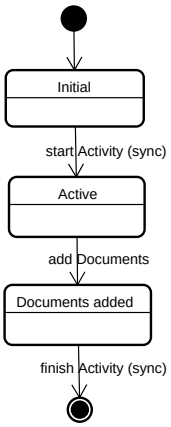

Fig. 17. Activity Information Automaton

To describe the whole picture, we present the *activity control aspect* and the *activity information aspect* automata in Fig. 16 and Fig. 17 respectively. These automata synchronize with the activity core automaton. The activity control automaton looks exactly like the activity core automaton, because there is no additional functionality in the control aspect except starting and finishing the activity. But the information aspect, for example, has to add documents to the started activity, it is described by the event "add Documents" in the automaton.

The overall scheme of all the automata and the ways of their synchronization is presented in Fig. 18. It depicts different types of automata and two dimensions of their interaction.

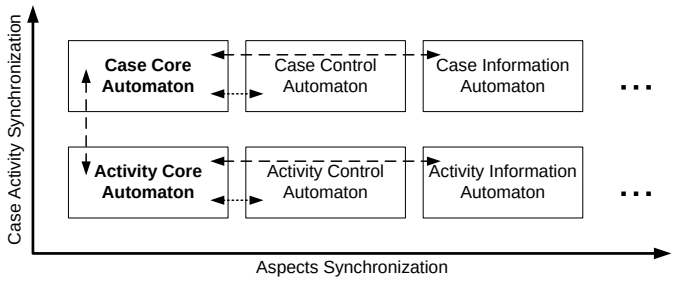

Fig. 18. The Overall Scheme of Automata and Synchronization

4.2 Architecture of the Workflow Engine

In this section, we briefly describe the extension of the *architecture of the workflow management system* in order to support the integration of new aspects and new formalisms. In Fig. 19, we present the Workflow Reference Architecture with a new component called *Aspects and Formalisms Definition* and a new interface between it and the workflow engine.

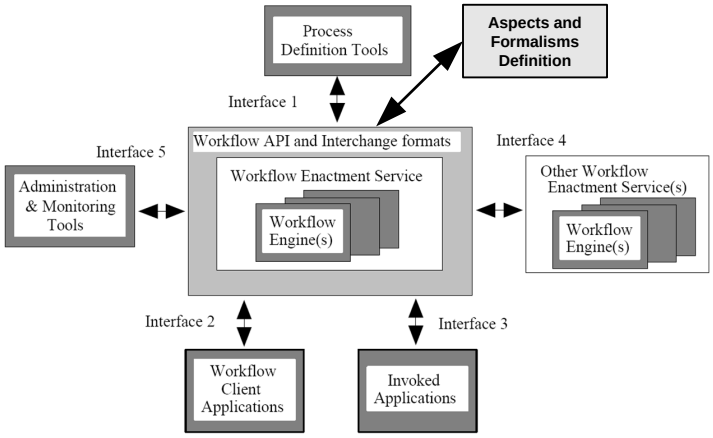

Fig. 19. WfMC Reference Model with the Aspects Interface

The aspects and formalisms definition component should be used by a developer and a designer in the following way:

- for introducing a meta-model for the aspect and integrating it with the existing ones;
- for inserting a formalism for a particular aspect and integrating the meta-model of the formalism to the meta-model of the aspect;

– for developing an editor for the formalism and integrating it with the existing editors for the other formalisms;
– for modelling and implementing the behaviour of the aspect and synchronizing it with the behaviour of the core and of the other aspects;

4.3 Implementation

In the previous sections, we have presented the meta-models for AMFIBIA. They covered the concepts of business process modelling, their relation as well as their dynamic behaviour in terms of automata. Based on these concepts, we have implemented a workflow engine. Actually, we simplified the meta-models in this paper, since we could not go into all the details of the actual meta-models of the implementation. In particular, the automata defining the behaviour and the interaction among the different aspects are more complex in the implementation – as mentioned earlier, the automata presented in this paper cannot execute activities of the same case concurrently. The automata used for the implementation take care of concurrency.

The meta-models were presented in an *Aspect-oriented Modelling* style, though we did not use a specific formalism: The basic concepts of business process modelling, the business process itself, the task, the case, and the activity are defined for each aspect separately[5]. The events defined for an aspect correspond to the *join points* of AoM, and the synchronization of the same event in the automata for different aspects correspond to *cross cuts*. The unsynchronized events resemble *actions*. The automata for the core of business process models corresponds to the *base program*. A technical difference, though, is that in most approaches to AoM, there is step of *weaving* which generates a single program that has code for all the aspects resp. concerns. Our approach rather executes the different automata independently of each other, and synchronizes the shared events. This, however, is an implementation issue only.

Since there is no mature software-technology for generating code from these models, we translated the concepts of Aspect-oriented Modelling to object-oriented models first. For example, for defining aspects of some concept such as a case or an activity there is an interface for registering aspects with a concept. Each class implementing this interface, can register with the concept which it is an aspect of, as was discussed in Fig. 7 already. This interface along with a composition relation is the object-oriented representation of aspects. These models where formalized in EMF, the Eclipse technology supporting UML. From these models, the basic Java classes of the workflow engine could be generated fully automatically. Moreover, we defined an interface for events and automata. Each aspects defines some events and registers with some of the events defined by other aspects so that the automata defining the dynamic behaviour of the aspects can be synchronized. Up to now, the automata are implemented directly

[5] Note that in most papers on Aspect-oriented Modelling, the term *aspect* is used on a more technical level. What we call aspect is often called a *concern* in AoM; but, we think that aspect is the better term.

in Java code. And there is a manager that coordinates the execution of the different automata belonging to all the aspects of a case: triggers their transitions and synchronizes the shared event by a two-phase-commit protocol.

In the future, we will develop a software technology, which supports the concepts of Aspect-oriented Modelling directly, so that we can generate the basic parts of the workflow engine from the AoM style meta-models as presented in this paper – without a detour to purely object-oriented models. Moreover, the automata defining the dynamic behaviour of the aspects could either be interpreted for implementing the workflow engine, or there could be translation to Java code. The development of the exact concepts and notations for these models as well as the technology for Aspect-oriented Modelling on top of EMF, however, is a long-term project in the field of software engineering; more details on this technology will be discussed in a separate paper addressed to software engineers. Moreover, we will also deal with a formal foundation of the interaction and sychronization of the automata defining the dynamic behaviour, e. g. in terms of process algebras.

Actually, there are some further implementation issues that are not covered in this paper, since they are more interesting from the software engineering point of view than from the business process modelling point of view. The workflow engine interacts with the agents of business processes via the work-list; moreover, it provides an interface for accessing the relevant documents, which is a kind of a light-weight application interface. Parts of these interfaces need information that comes from different aspects. In order for AMFIBIA to be fully aspect independent, the parts of an aspect that are relevant to the graphical user interface for the agents need to be implemented with each aspect. Our implementation takes care of this, but we do not discuss the details here.

5 Related Work

Wil van der Aalst and Kees van Hee, in their book about Workflow Management [37] present a reference framework for defining the business processes. They define the business-process management context of workflow management systems. The authors define an ontology, where "business process management" is the domain of interest. They deal with the processes, management of the processes and information systems related to them. This work serves as a background of the current paper, which aims to develop a meta-model, which requires usage of more formal techniques and technologies.

The book of Leymann and Roller [29] discusses the basics of the workflow technology and its aspects, the models of business processes, and the basics of workflow management systems. The book also discusses the meta-models and constructs that have to be provided to model their environment. The concepts and necessary terms in the area of workflow management are presented for the process model, organizational model, versioning model and others. One part of this book precisely describes the fundamental part of the meta-model, which is very close to the implemented one in MQSeries Workflow. The syntax and

the semantics of the meta-model are described using process model graphs and elementary operations of the set theory. The book influences significantly this paper, since it presents the definitions of the meta-model constructs with precise syntax and semantics. Therefore, this meta-model can be instantiated to the modelling level and used for the formal workflow modelling.

In the book of Jablonski, Böhm and Schulze [24] also the topic of meta-modelling is discussed. The meta-model or "metaschema", as it was called by the authors, describes the properties of all the products of the workflow language. The book presents the schema of the meta-levels and describes relations between them. In the other work of Jablonski [25], it is analysed how workflow management can support enterprise application integration tasks and e-commerce applications.

The book of Scheer describes the Architecture of Integrated Information Systems (ARIS) and their approach for integrating different business process modelling aspects [35]. They divide the general business process model context into views: organization, data, control and function.

Zur Muehlen and Rosemann [34] analyse the usage of the workflow meta-models in particular workflow management systems. They present the ideas of evaluation and comparison of meta-models of different tools, such as WorkParty and FlowMark. An important result of their paper is the necessity of independent reference models, which could serve as the evaluation benchmark of meta-models.

The Organizational Structure Facility (OSF) Specification [1] from OMG presents an organizational meta-model in MOF compliant form, illustrated as a set of UML diagrams.

The Workflow Reference Model of WfMC [20, 22] provides a common vocabulary for describing a business process and its aspects, functional description of software components of the workflow management system and interface between them. The "Reference Model" provides a framework supplying different specifications of the workflow management systems to be developed within a common context. The Reference Model also defines the object technology as a possible target implementation model for workflow systems. A discussion paper of WfMC about the common object model [21] can be regarded as a proposal for usage of object-oriented technology on the level of implementation for workflow.

In XML Process Definition Language Specification of WfMC [4], the meta-model of the process definition, containing the main entities, their relationships and attributes, was defined. This meta-model could be used for exchange of process definitions. The textual description of semantics of these entities is also provided.

In the Workflow Management Facility Specification [2], OMG and WfMC joined together to define a set of IDL interfaces for introducing of the workflow technology to the OMG architecture. They defined the interfaces for workflow execution control, monitoring, and interoperability between workflows. This interface model is actually a UML class diagram and is specified by IDL interfaces. The specification is actually done on the modelling level (comparing to MOF M1 layer) and can be instantiated to particular objects.

In the work about conceptual modelling of workflows [10], the authors try to make an effective convergence between workflow management and databases by means of formalization of specification and providing a language for workflow applications at the conceptual level.

Karagiannis and Kühn present the meta-modelling concepts and a generic architecture for the meta-modelling platforms in their paper [26]. This work also includes three best practise examples from the industry.

The other part of the work, related to the current one, is dealing with the enterprise process modelling. For example, the work "Toward and Integrated Framework for Modeling Enterprise Processes" [12] presents the motivation and concepts of using the Enterprise Process Modelling Language (EPML), which builds on IDEF, DFDs and EPCs, for modeling the enterprise processes, including different aspects of business process management. This specifications in this visual language can be converted to the Petri net models, and thus, have formal semantics.

The other area, which is tightly related to the dynamic interaction of aspects described in Sect. 4, is the area of aspect-oriented programming and aspect-oriented modelling. The works of Rémi Douence et al. [13, 14] discuss formal automata-based approaches for aspect-oriented modelling of program executions. Two general surveys on aspect-oriented analysis and design and on languages and models [11, 9] serve as a background knowledge in this area.

In this Section, we have enumerated a set of different formalisms for defining the meta-models and the ontologies for business processes. Since there are different aspects of the business process management covered by one or the other mentioned approaches, we have defined a meta-model, which contains the independent meta-models for different aspects but can also integrate the meta-models. This meta-model is not only on the conceptual level but also prescribes the future implementation. Therefore, the meta-model fits to the concept of the Model-Driven Architecture (MDA), and, consequently, is implemented in a workflow engine. Since there exists a variety of meta-modelling approaches, we have defined a general formalism-independent meta-model, which can prescribe a common exchange format between them. Such an exchange format can be defined using XML Metadata Interchange (XMI), cause the mapping from MOF to XMI is defined in the OMG specification. The implementation of the dynamic interaction of aspects proceeds from the ideas of aspect-oriented modelling and from introducing and adapting these ideas to the business process modelling domain.

6 Conclusion

In this paper, we outlined the ideas of AMFIBIA, a meta-model for business process modelling, which captures not only the aspects, concepts and their relation, but also the dynamic behaviour and the interaction among the different aspects. The main justification of yet another 'ontology' for business process modelling

is that it covers the dynamics of such systems. Moreover, AMFIBIA is more ambitious than existing 'ontologies', architectures, frameworks or meta-models:

- AMFIBIA shows that, conceptually, the different aspects of business processes can be modelled independently of each other and that it is possible to integrate them via the integral parts.
- AMFBIA is open for additional aspects and is not biased towards any aspect.
- AMFIBIA is independent of any particular modelling formalism and, actually, defines an interface that can be implemented by most formalisms for that particular aspect.

The implementation of a prototype workflow-engine based on AMFIBIA shows that its concepts really work.

Acknowledgement We would like to thank Elżbieta Pielenz and Ranghild van der Straeten for many discussions on business processes and on Aspect-oriented Modelling, which eventually resulted in the ideas presented in this paper.

Moreover, we thank the members of the 'Projektgruppe AMFIBIA' at the University of Paderborn for implementing the concepts of AMFIBIA in a one-year master's project. During this implementation, we had many discussions which helped to clarify and refine the concepts of AMFIBIA, and to make it run. The members of the 'Projektgruppe AMFIBIA' are: Achim Heynen, Andrè Altenau, Christiane Klapdohr, David Schmelter, Dennis Goeken, Elmar Köhler, Patrick Könemann, and Peter Pietrzyk.

References

1. Organizational Structure Facility (OSF) specification. Technical Report bom/2000-02-03, Object Management Group, February 2000.
2. Workflow Management Facility specification, v1.2. Technical report, OMG, April 2000.
3. Capability Maturity Model Integration (CMMISM), Version 1.1. Technical Report CMU/SEI-2002-TR-012, Carnegie Mellon, Software Engineering Institute, March 2002.
4. Workflow Process Definition Interface – XML process definition language. Technical Report WFMC-TC-1025, Workflow Management Coalition, October 2002. Version 1.0.
5. Björn Axenath, Ekkart Kindler, and Vladimir Rubin. An open and formalism independent meta-model for business processes. In E. Kindler and M. Nüttgens, editors, *Workshop on Business Process Reference Models 2005 (BPRM 2005), Satellite event of the third International Conference on Business Process Management*, pages 45–59, September 2005.
6. J. Becker and P. Delfmann, editors. *Referenzmodellierung*. Physica-Verlag, 2004.
7. Jörg Becker, Michael Rosemann, and Christoph von Uthmann. Guidelines of business process modeling. In *Business Process Management, Models, Techniques, and Empirical Studies*, pages 30–49, London, UK, 2000. Springer-Verlag.
8. Philip A. Bernstein, Vassos Hadzilacos, and Nathan Goodman. *Concurrency Control and Recovery in Database Systems*. Addison-Wesley, 1987.

9. Johan Brichau and Michael Haupt. Survey of aspect-oriented languages and execution models. Technical Report AOSD-Europe-VUB-01, AOSD-Europe, May 2005.
10. F. Casati, S. Ceri, B. Pernici, and G. Pozzi. Conceptual modeling of workflows. In M. P. Papazoglou, editor, *Proceedings of the OOER'95, 14th International Object-Oriented and Entity-Relationship Modelling Conference*, volume 1021, pages 341–354. Springer-Verlag, 1995.
11. Ruzanna Chitchyan, Awais Rashid, Pete Sawyer, Alessandro Garcia, Mónica Pinto Alarcon, Jethro Bakker, Bedir Tekinerdogan, and Andrew Jackson Siobhán Clarke and. Survey of aspect-oriented analysis and design approaches. Technical Report AOSD-Europe-ULANC-9, AOSD-Europe, May 2005.
12. Nikunj P. Dalal, Manjunath Kamath, William J. Kolarik, and Eswar Sivaraman. Toward an integrated framework for modeling enterprise processes. *Commun. ACM*, 47(3):83–87, 2004.
13. Rémi Douence, Olivier Motelet, and Mario Südholt. A formal definition of crosscuts. *Lecture Notes in Computer Science*, 2192:170–184, 2001.
14. Rémi Douence and Jacques Noyé. Towards a concurrent model of event-based aspect- oriented programming. In *European Interactive Workshop on Aspects in Software (EIWAS 2005), Brussels, Belgium*, 2005.
15. Joerg Evermann and Yair Wand. Towards ontologically based semantics for UML constructs. In *Proceedings of the 20th International Conference on Conceptual Modeling*, pages 354–367. Springer-Verlag, 2001.
16. P. Fettke and P. Loos. Referenzmodelle für den Handel. *HMD - Praxis Wirtschaftsinform.*, 235, 2004.
17. P. Fettke and P. Loos. Der Beitrag der Referenzmodellierung zum Business Engineering. *HMD - Praxis der Wirtschaftsinformatik*, (241):18–26, 2005.
18. Jim Gray and Andreas Reuter. *Transaction Processing: Concepts and Techniques*. Morgan Kaufmann, 1993.
19. David Harel and Bernhard Rumpe. Modeling languages: Syntax, semantics and all that stuff. Technical report, The Weizmann Institute of Science, Rehovot, Israel, MCS00-16, 2000.
20. David Hollingsworth. The Workflow Reference Model. Technical Report TC00-1003, The Workflow Management Coalition (WfMC), January 1995.
21. David Hollingsworth. A common object model discussion paper. Technical Report WfMC-TC-1023, WfMC, March 1999.
22. David Hollingsworth. The Workflow Reference Model: 10 years on. Technical report, WfMC, 2004.
23. International Standard ISO/IEC. Software and systems engineering – High-level Petri nets, part 2: Transfer format. Technical Report 15909-2. Working Draft Version 0.9.0, ISO/IEC, June 2005.
24. Stefan Jablonski, Markus Böhm, and Wolfgang Schulze. *Workflow-Management Entwicklung von Anwendungen und Systemen*. dpunkt.verlag, 1997.
25. Steffen Jablonski. Workflow management as an integration platform. In *Proc. of 2nd Int. Coll. on Petri Net Technologies for Modelling Communication Based Systems*, pages 135–138, 2001.
26. D. Karagiannis and H. Kühn. Metamodeling platforms. In K. Bauknecht, A.Min Tjoa, and G. Quirchmayer, editors, *Proceedings of the Third International Conference EC-Web*, volume 2455 of *LNCS*, page 182. Springer-Verlag, September 2002.
27. E. Kindler. Using the Petri Net Markup Language for Exchanging Business Processes? Potential and Limitations. In M. Nüttgens and J. Mendling,

editors, *XML4BPM 2004, Proceedings of the 1st GI Workshop XML4BPM – XML Interchange Formats for Business Process Management at 7th GI Conference Modellierung 2004, Marburg Germany, March 2004*, pages 43–60, http://wi.wu-wien.ac.at/~mendling/XML4BPM/xml4bpm-2004-proceedings-pnml.pdf, March 2004.
28. J.C. Laprie. *Dependability: Basic Concepts and Terminology in English, French, German, Italian and Japanese*, volume 5 of *Dependable Computing and Fault Tolerant Systems*. Springer, 1992.
29. Frank Leymann and Dieter Roller. *Production Workflow: Concepts and Techniques*. Prentice-Hall PTR, Upper Saddle River, New Jersey, USA, 1999.
30. Meta Object Facility (MOF) specification. Technical report, Object Management Group, April 2002.
31. M. Nüttgens and J. Mendling. XML4BPM 2004, proceedings of the 1st GI workshop XML4BPM – XML interchange formats for business process management at 7th GI conference modellierung 2004, marburg germany, march 2004. http://wi.wu-wien.ac.at/~mendling/XML4BPM/xml4bpm-2004-proceedings-pnml.pdf, March 2004.
32. Woody Pidcock. What are the differences between a vocabulary, a taxonomy, a thesaurus, an ontology, and a meta-model?, 2004. Web Page URL: http://www.metamodel.com/article.php?story=-20030115211223271.
33. Jan H.P. Eloff Reinhardt A. Botha. A security interpretation of the Workflow Reference Model. In *Information Security - from Small systems to management of secure infrastructures*, volume 2, pages 43–51, August 1998.
34. Michael Rosemann and Michael zur Muehlen. Evaluation of workflow management systems - a meta model approach. In Keng Siau, Yair Wand, and Jeffrey Parsons, editors, *2nd CAiSE/IFIP 8.1 International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD '97)*, Barcelona, 1997.
35. A.W. Scheer. *Architecture of integrated information systems - bases for company modeling, 2nd edition*. Springer, Berlin, 1992.
36. A.W. Scheer. *Wirtschaftsinformatik. Studienausgabe. Referenzmodelle für industrielle Geschäftsprozesse*. Springer, Heidelberg, 1997.
37. Wil van der Aalst and Kees van Hee. *Workflow Management: Models, Methods, and Systems*. Cooperative Information Systems. The MIT Press, 2002.
38. Michael Weber and Ekkart Kindler. *The Petri Net Markup Language*, pages 124–144. LNCS 2472. Springer, 2003.
39. Mathias Weske, Thomas Goesmann, Roland Holten, and Rüdiger Striemer. A reference model for workflow application development processes. In *WACC '99: Proceedings of the international joint conference on Work activities coordination and collaboration*, pages 1–10, New York, NY, USA, 1999. ACM Press.
40. Workflow Management Coalition: Workflow security considerations - white paper. Technical Report WFMC-TC-1019, The Workflow Management Coalition (WfMC), February 1998.
41. Workflow Management Coalition: Terminology & glossary. Technical Report WFMC-TC-1011, The Workflow Management Coalition (WfMC), February 1999.
42. Wenming Wu and Yisheng Dong. Metamodeling-based Semantic Web languages. In *EC-Web*, pages 339–347, 2003.
43. Michael zur Muehlen. Organizational management in workflow applications. *Information Technology and Management*, 5(3):271–291, 2004.