Chapter X
A Design Tool for Business Process Design and Representation

Roberto Paiano
Università di Lecce, Italy

Anna Lisa Guido
Università di Lecce, Italy

ABSTRACT

In this chapter the focus is on business process design as middle point between requirement elicitation and implementation of a Web information system. We face both the problem of the notation to adopt in order to represent in a simple way the business process and the problem of a formal representation, in a machine-readable format, of the design. We adopt Semantic Web technology to represent process and we explain how this technology has been used to reach our goals.

INTRODUCTION

Today, the impact of business processes within companies gains more and more importance and provides tools to the managers, and methodologies useful to understand and manage them are a must. It is important to integrate business processes in the overall information system (IS) architecture with the goal to provide, to the managers, the right flexibility, to avoid the reimplementation of the applications in order to follow the business process changes, and to adapt the existing applications to a different management of the existing business logic. As a consequence the process-oriented management requires both the ability to define processes and the ability to map them in the underlying system taking into consideration the existence of heterogeneous systems.

It is clear that business and information technology (IT) experts must work together to provide the right flexibility to the IS and thus to improve the overall management. The semantic

gap between business and IT experts is a problem in the development of an overall system oriented to the improvement of the business process. The first thing to do to solve this problem is to study a common language between these two classes of users with very different requirements:

- Business experts will focus on the processes and on the direct management of them in order to modify the company work without giving many technical details: A change to the process must be immediately translated in a change to the applications that implement it.
- IT experts require more details about processes and require simplicity in order to understand the process flow and thus application requirements.

Business process design is the middle point between requirement elicitation and Web IS implementation that is between business and IT experts. The tool that supports business process design must be the same for both business and IT users and must answer to two different key aspects:

- Easy to use with a notation easy to understand and allows to gives all technical details but, at the same time, hiding the complexity to the final user.
- Supports the export of the process design in a formal way in order to give to the IT experts all the process detail that they need. The process description must be machine readable.

In our research work we consider these two aspects by two approaches:

- The use of a standard notation for business process representation.
- The use of an ontological language that, thanks to its flexibility and machine-readable feature, is able to express all process complexity in a formal way.

In the next section of this chapter we explain the background about the concept of business process management (BPM) and the analysis of several BPM suites, and then we explain the open issue and the possible solutions related to the BPM suites. Next, we present our approach to the business process representation and we provide an overview about business process management notation (BPMN), Semantic Web languages, and about the concept of the metamodel. In the next section we explain what metamodel means and what the main problems are in the meta object facility (MOF) approach. In the section: *BPMN Ontological Metamodel: Our Approach to Solve MOF Problems* we explain how to solve problems about the classical metamodel approach with the use of the Web Ontology Language (OWL) and then, in the next section, we explain the steps followed to develop the ontological metamodel. Finally, we highlight the future trends about our research work and the conclusions.

BACKGROUND

Since the 1990s, process management has gained more and more importance in companies. Process management began to affirm with the *business process reengineering (BPR) theory* (Hammer, 1990) that allows us to improve company management thanks to the analysis and redefinition of the company's processes. BPR theory does not give a structured approach to the introduction of the process in the overall IS architecture but the process logic was in the mind of IT experts that were free to implement them.

Starting from BPR, the evolution was workflow idea (proposed and supported by Workflow Management Coalition [http://www.wfmc.org]), which is the automation of some companies' processes where only people performed process steps.

BPR theory and workflow idea allow to take into consideration vertical processes involving a single company department, but process, usually, covers several departments (a process may involve also several companies), so BPR and workflow do not cover the overall company complexity.

The traditional vertical vision of the company that locates business logic in functional areas is not the best way to provide the overall vision of the company and to improve process management, so this vision is today abandoned: Managers need a horizontal vision with the goal to manage the overall flow and to understand and correct—in the most rapid way possible—managements errors. Obviously, we speak about administrative business processes extended throughout the entire organization where they take place.

Actually, the attention is turned to the BPM with an innovative idea that allows us to have a horizontal (rather than vertical) vision of the company and to consider processes where steps are performed both from people and systems. The goal of BPM is to make explicit the process management in the IS architecture. Process logic is, today, hidden in the application level and, often, the way with which this is made is in the mind of the IT experts of the company, and it is not well documented. The IS obtained is difficult to maintain. A change to the process logic needs a revision to the business logic, and this requires a lot of time and incurs high costs. The processes are not explicit and thus it is very difficult to monitor and manage them.

There are a lot of BPM suites on the market that try to make explicit the definition and management of the processes. A recent study (Miers, Harmon, & Hall 2005) compares different BPM suites (http://www.Fuego.com; http://www.ilog.com; http://www.popkin.com; http://www.w4global.com; http://www.filenet.com) from different points of view such as cost, platform, user interface, and so forth. These BPM suites allow us to represent processes and to manage their execution; and they provide administration tools that allow us to manage processes and users involved in the system. Several suites focus on document management and thus on the possibility to manage, with a high degree of security, documents that are involved in the process, versioning, and so forth. BPM suites also provide a Web application that allows different actors in the process to load their task and to work with it. In the Web application a single user can view and work with the process and, despite this being visible in the Web, each user has access to only one page and thus information, navigation, and transactional aspects of Web application are not taken into consideration. An important aspect of the BPM suites is the application program interface (API) that allows us to develop a custom application but does not supply any information about design.

BPM SUITES: OPEN ISSUE AND POSSIBLE SOLUTIONS

The main problems highlighted in the study of these suites are:

- The high cost of suites are difficult to apply in small- to medium-sized companies, and they require high investments both to purchase hardware and to train people in the company that will use these frameworks.
- Ad hoc notation to represent processes that are often hard to read and to understand both from business experts and from IT experts and thus the semantic gap problem is not solved.
- There is a lack of methodology that helps in the transition from process design to the final Web application.

In this study we focus on two of these main problems: high costs and process representation.

Small- to medium-sized companies may have several advantages from BPM suite because it

helps to control all the company activities in a simple and easy way. Small- to medium-sized companies have several economic problems in acquiring these suites, a part from hardware and software costs there is the necessity to reach skilled people able to work with these complex suites. The difficulty to acquire hardware, software, and people make it impossible for these companies to adopt the BPM suite and, of consequence, to improve their overall management.

In regards to a low-cost BPM suite, there is another important problem: BPM suite may be used both from IT and from business people but business people have a technical experience about management and they do not understand IT technical aspects; IT experts do not know management aspects. The semantic gap is very large and hard to cover, so it is important to take into consideration only notations that are easy to learn both by IT and business experts. The business process notation must be, of consequence, simple, easy to use, and easy to understand both by business experts and by IT experts, in a few words, the notation to represent processes must cover the semantic gap between business and IT experts.

There is a different way to represent business process. Unified modeling language (UML) activity diagram, for example, allows a defining process but it is not simple to understand: As an example, the actor of the processes is not immediately visible. The same problem is true for the traditional workflow representation, that is, it is not intuitive and allows defining only the process flow without taking into consideration human and/or system interaction. UML, standard de facto in the software analysis, may be useful for IT experts but hard to learn, to use, and to understand for business experts; workflow may be, instead, useful for business experts but hard to understand for IT experts.

Exploring several notations, we study a recent notation (White, 2004) proposed by the business process management initiative (BPMI) that, thanks to its simplicity and completeness, seems the best way to represent a process.

BPMN, today, is not a standard but it is supported by several companies, and it does not allow designing information, strategies, or business rules.

The design obtained through BPMN is clear and it is easy to understand which actors (human or system) are involved in the process and what the relationships are between them.

To complete a BPM suite the graphical process representation is not enough to automate the business process so we need a formal representation of the process.

Starting from BPMN notation (and from its complexity) we observe that there is not a well-defined, machine-readable standard to represent a process (business process execution language [BPEL]; business process execution language for Web services [BPEL4WS], and so forth). From these considerations, the final goal of our study is to develop a light framework to manage processes. This framework will be efficient, easy to use, and low cost. In this phase of our study we focus on the design and implementation of a business process editor and we face two main problems: (1) the choice of the notation to adopt in the business process design, and (2) the choice of the formal language to adopt in order to make the design machine readable.

BUSINESS PROCESS REPRESENTATION: OUR APPROACH

The first and most important problem to solve to reach the goal to define a low cost framework that supports process definition and management is to select the notation to adopt. The notation must cover the semantic gap between business and IT experts and must answer to two main requirements: completeness and simplicity. These aspects may be found in the BPMN notation: Its main

goal is to cover the gap between IT and business experts, which is the gap between process design and process implementation. BPMN notation is, also, very easy to learn and to understand, so we select it as the notation to represent processes.

As we saw in the previous section, BPMN notation is not tied to a specific machine-readable format but there are several machine-readable formats not yet standard. In our research work, we explored several alternatives before choosing a language to represent processes; finally, we chose to use ontology *in an innovative way*: Our use of ontology is different from the traditional Semantic Web where ontology describes, in general, a domain of knowledge. We adopt both the ontology and the concept of metamodel: Ontology is, in our research work, the language to represent both the BPMN metamodel (that we develop) and the process model starting from the metamodel. The ontological language used in our research work is OWL (World Wide Web Consortium, 2004a): a text language without graphical notation.

To understand the following part of our study we introduce an overview about BPMN notation, next a brief definition of ontology and of the semantic languages, and finally, we present the concept of metamodel.

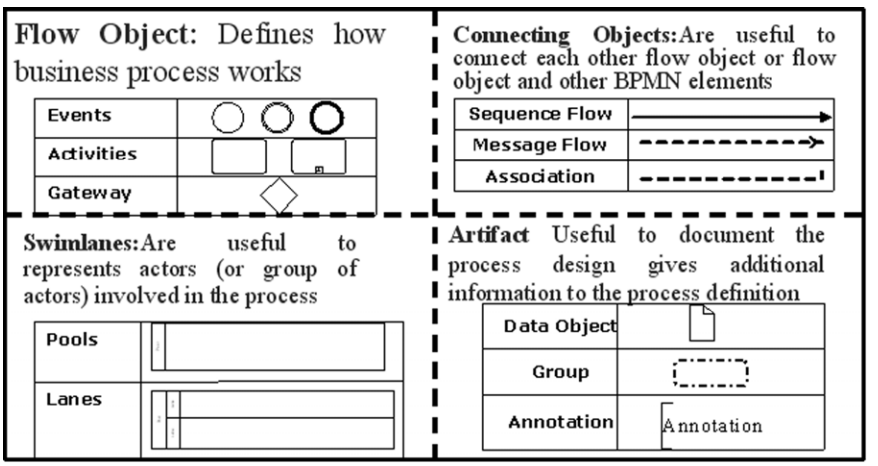
BPMN Notation Overview

In daily BPM, business experts start their analysis with the design in the large of the process: Details are given to the design in the following steps. Implementation details, that are details needed in the implementation phase, are given in the last step of the analysis and are obviously given by IT experts and not by business experts. To follow this natural evolution from design in the large to design in the small, BPMN notation is made up of two different levels of details:

- A core object (business process diagram modeling objects) made up of base elements that allow us to define a process in the large.
- An extension mechanism that allows us to extend core objects and to add properties to obtain a detail level near to the detail needed in the implementation phase.

These different detail levels make the final design easy to understand not only by experts of the notation but also by nonexperts of the notation and thus by IT experts that may operate directly in the design by adding their details. So, in the

Figure 1. Main primitives of BPMN notation



design phase both IT and business experts may provide all the details needed.

Business process diagram modeling objects are made up of four different groups of primitives: Flow Object, Connecting Object, Swimlanes, and Artifact (Figure 1).

Flow objects have three types: *Events* that represent something that occurs during the normal process execution; events have a cause (trigger) or an impact (result). Different icons allow us to define start, intermediate, or end an event. Internal market (and this is another detail level) represents triggers that define the cause of the events. For example, start event may be a circle without any icon or may have an envelope icon to represent that the process starts when a message is arriving. A complete set of events and icons are shown in Figure 2.

Another type of Flow Object is *activities,* which is generic work that a company performs. Activities may be of two different types: *task* and *sub-process*. Task is an atomic activity, that is, the work not broken down to a finer level of process model detail; sub-process is an activity made up of several subactivities.

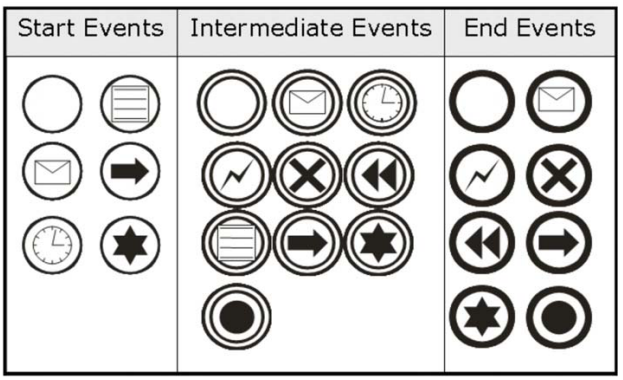
The third type of Flow Object is the *gateway* used to control the divergence and convergence of multiple sequence flow. Gateway represented with a diamond has, as events, different icons to represents different possible flow control behavior.

Flow objects may be connected to each other or may be connected to Artifact by three different types of connecting objects: *sequence flow* is used to connect a Flow Object with the next Flow Object that will be performed in the process. The second Connecting Object is the *message flow* used to show the flow of messages between two participants (participants are represented by Swimlanes); finally *associations* are used to connect information (Artifact) with Flow Object.

Swimlanes are used to group Flow Object, Connecting Object, and Artifact. Swimlanes are made up of Pools that represents a participant in a process. A Pool may be subpartitioned by lanes used to organize and categorize activities. For example if the designer wants to represent the activity in an office and to highlight the role of each person in the process, it is possible to use a Pool to represent the office and, within the Pool, lanes will be used to represent each person.

Artifact may be used for documentation purposes and does not have direct effect on the normal process flow. Artifact may be of three different types: (1) *data object* that is often used to represent a document required and/or produced by an activity; (2) *group* is often used to identify activities in different Pools, and finally (3) *text*

Figure 2. Events icon defined in BPMN notation



annotation helps designers to give additional (textual) information to the reader of the design.

In order to have a final diagram easy to understand, BPMN notation defines connection rules for both sequence flow and for message flow. Connection rules are useful to understand how Flow Object, Swimlane, and Artifact may be connected to each other, and what condition is required to connect them together. As an example, message flow cannot connect objects in the same Lane but only objects in two different Lanes; similarly sequence flow cannot connect an object in two different Lanes but only objects in the same Lane.

At this point we can observe that with four different types of objects (and relative subtypes) and following simple connecting rules, business users are able to design the process in all its complexity, but implementation details are needed.

BPMN defines another detail level: each BPMN elements has its own properties. Suppose, for example, that the designer defines a start event of type "message." In the implementation phase IT experts need to know what message is required and what technology is used to send/receive the message. Start event has among its property the attribute message that allows us to supply the message to send/receive and the attribute implementation to define the technology used to receive the message (Web services or other technology).

Other details about property are out of the scope of this work but will be found in the BPMN specification. Here we want to underline all the BPMN complexity and the different level of detail that composes this notation; thanks to this different details level it is possible to use the same notation both for business and for IT people.

What is Ontology?

The traditional philosophic concept of ontology, that is, "a systematic explanation of being," has been inherited in the artificial intelligence (AI) with several definitions. In the AI idea, ontology is the study of something that exists or may exist in some domain; ontology may be defined as terms linked together to define a structure in a well-defined application domain.

The definition of ontology near to the IT aspect is given by Gruber (1993, pp. 199-220): "ontology is a formal, explicit specification of a shared conceptualization." The concept of conceptualization is the abstraction of some concept through the definition of some peculiar characteristic; the term *explicit* is related to the fact that constraints about the concept must be defined in an explicit way; finally, *formal* means that ontology must be defined in a machine-readable format.

Ontology is a collection of terms and related definitions or a map of concepts where we can forward or backward from one concept to another in a well-defined domain.

The application of the ontology in the Semantic Web was defined by Berners-Lee, Hendler, and Lassila (2001) as "the semantic web is not a separate web but an extension of the current one in which information is given well-defined meaning, better enabling computers and people to work in cooperation." The necessity to go over traditional Web and to go into the Semantic Web is due to the fact that the Web contains a tremendous amount of data, information, and knowledge freely available but poorly organized: A large amount of information is in textual format, thus it is difficult to filter and to extract content.

Traditional Web works on information retrieval are based on keyword search and on manual classification of the content: To reach information on the Web we need to write a keyword on a search engine and thus to manually filter between all results to reach the right result nearest to our goal. Semantic Web is oriented to the semantic information retrieval, and on a machine-machine cooperation aimed to select the right information based on the concept behind the keyword. The goal of the Semantic Web is to make explicit the knowledge and to integrate different sources of

knowledge with the goal to extract knowledge from knowledge.

Semantic Web Languages

Behind the idea of Semantic Web, the World Wide Consortium (W3C) works around languages for knowledge representation. One of the main goals of the ontology is the interoperability of both syntactic and semantic. Semantic interoperability means that ontology must be machine readable, that is, it must be interpreted by a machine in a well-defined format. Syntactic interoperability is the ability to provide support to a reason that is to learn from the data. Languages born to support ontology are different: The first ontology language is resource description language (RDF) (W3C, 2004c) and RDF schema (RDFS) (W3C, 2004b).

RDF has a model similar to the entity-relationship model which allows us to give interoperability through applications that interact with each other in order to exchange information on the Web in a machine-readable format. RDF does not give reasoning support but has the basis to achieve this. The three main concepts of RDF are:

- **Resource.** Anything that will be defined in RDF is named Resource. Resource is named by a uniform resource identifier (URI) and can be a Web page or part of it; a resource can be an object in the real word not directly accessible on the Web.
- **Property.** Property allows us to define a characteristic of a resource through a binary relationship between two resources or between a resource and a well-defined data type.
- **Statement.** It is a sentence with a fixed structure: subject, predicate, and object. Subject and predicate must be a resource while an object may be a literal. Statement allows us to represent complex situations if the object is used as a subject on a new statement.

Successors of RDF (and RDF schema) are Darpa Agent Markup Language (DAML) and Ontology Interchange Language (OIL); these two languages are antecedent to the Ontology Web Language (OWL) used today.

OWL allows us to provide more machine readability than extensible markup language (XML), RDF, and RDF schema. In the Semantic Web, OWL is used when information must be processed from application (and not only presented to human). OWL allows us to provide a detailed description of any domain.

OWL added new vocabulary (compared to RDF and DAML+OIL) to describe classes and relationships; it supports a useful mechanism to integrate different ontologies. OWL is made up of three different languages each of them is the extension of its ancestor:

- OWL lite allows us to give simply a taxonomy without complex constraint.
- OWL description logic (OWL DL) gives high complexity, completeness, and decidability.
- OWL full gives expressiveness but does not give decidability.

The OWL main primitives are:

- **Classes.** Classes allow the abstraction of some concepts. Each class has a set of properties (each one for specific concept characteristics). A class would be composed by subclasses.
- **Properties.** There are two types of properties: DataType specific to each class and ObjectProperty used to create a link between classes. ObjectProperty has both domains:

Figure 3. MOF and ontological approaches compared

Approach / Level	MOF	Ontological
Meta-meta model (M3)	MOF-language	OWL-language
Meta-model (M2)	Classes, associations, packages	Ontological classes and properties
Model (M1)	Derived classes, associations, packages	Instances of classes and properties
Data (M0)	Data	Data

class (to which the property is connected) and range (the possible values of the property). In each class we can indicate "restrictions" that define constraints.

- **Individuals.** Individuals are objects with the characteristics defined by classes and properties. Both classes and properties may have individuals.

What is Metamodel?

The metamodel idea was born about 10 years ago and the interest around it is increasing. A metamodel can be defined as a language to describe a model, so, to create a metamodel it is important to work in an abstract level. Metamodel allows us to describe a well-defined methodology, so, starting from the metamodel, it will be possible to define a model: metamodel provide, in a few words, guidelines for model definition.

The introduction of the metamodel idea has brought forth the introduction of metacase tools. Standard case tools support a fixed notation hard coded in tools: A change in the methodology requires a change in the code of the tool and this fact requires high costs and a lot of time. Metacase tools, based on the metamodel, allow us to separate notation from the methodology definition, so a change in the methodology will reflect in the tool with a few changes (or without change) in the code. To be included in the metacase tool a metamodel must be well defined so it must answer to three important requirements: a metamodel must be:

- **Complete.** It must cover all the primitives of the methodology that represent.
- **Extensible.** Metamodel must follow the methodology evolution so it will be possible to adapt the metamodel in a simple way without redefining it from scratch.
- **Semantic.** Metamodel must express all the semantics of the methodology primitives in order to give the right meaning to each element.

To avoid confusion between metamodel and model, we explain the meta-object facility (MOF) approach to meta-model proposed by the Object Management Group (OMG) (http://www.omg.org). MOF architecture is very helpful because it allows us to separate, in a simple way, the concept of the meta-model from the concept of the model: A model will be an instantiation of the meta-model.

MOF approach is based on a 4-level architecture. It allows us to define a language for the methodology representation and to use this language for model definition. The 4-level architecture proposed by OMG is very helpful to separate different levels of abstraction.

As show in Figure 3 in the M3 level (the meta-meta model level) the MOF language, that is, the abstract language used to describe MOF metamodel, is defined. In the M2 level MOF approach allows us to define the metamodel. MOF is object oriented and strictly connected to UML: UML notation is used to express MOF metamodel. The main MOF elements are classes, associations, and packages; moreover, to express model rules it is necessary to define constraints. MOF does not force the use of a particular language but suggests the object constraint language (OCL) (OMG, 1997). Starting from the metamodel defined in the M2 level, the designer of a particular methodology using metamodel (guidelines for methodology) designs its model. Finally M0 level represents data of a specific model.

MOF METAMODEL: OPEN ISSUE

The architecture proposed by OMG is very helpful to obtain a meta-model of BPMN notation, but in our study we highlight some problems related to the language in the M3 level strictly related to UML that impose some limits when used to define metamodel.

The first problem is about the *metamodel semantics*: It is very important to assign a meaning to every metamodel concept in order to have the meaning of each methodology primitive. In MOF approach the use of stereotypes to define primitives which are not directly supported by UML is intensive: A lot of primitives are not directly supported by UML and, thus, all primitives are represented by stereotypes. Metamodel semantics, consequently, coincide with stereotype semantics. Furthermore, the *lack of semantics* creates confusion to the unskilled designer during the practical applications of modeling concepts. The explicit presence of semantics helps the designer to understand how the modeling concepts should be used.

Another problem strictly connected to semantics concerns *semantic relationships among classes*: MOF allows us to use only two relationships: aggregation and association. In the definition of a metamodel methodology it is necessary to define specific methodology relationships (different from association and aggregation) with its relative semantics.

Another problem is that *relationships among classes* are lost in the transition from metamodel to model. Supposing that in the metamodel we have a relationship among classes: When we define the model, relationships among classes must be redefined because relationships are not inherited by the model. This problem could be solved creating intermediate classes to represent the relationships; the disadvantage of this solution is that it will make the model unreadable for the large number of intermediate classes.

Finally, in the MOF approach, each class has specific primitive attributes. If an attribute is the same for two different concepts, in MOF approach it is defined once for each class because each attribute is strictly connected to each class. This MOF limit creates confusion letting designers think that semantics are different for each attribute, while semantics are the same.

Another problem is the metamodel *flexibility,* that is, the possibility to enrich the model with new primitives defined in methodology or to add new characteristics to the primitives already defined. The solution proposed by UML (both 1.x and 2.0 [OMG, 2001; OMG, 2003]) is to enrich the UML metamodel with the extension mechanism. The *extension mechanism approach* is based on a good knowledge of UML. Another problem related to the language evolution concerns the unique name assumption principle: In the UML approach different words must refer to different objects. In

order to meet methodology evolution, it is often necessary to define new versions of concepts (defined before) and to use the same name. The unique name assumption makes it impossible. The UML and MOF do not support the *dynamic classification of classes*. It is possible that, when metamodel is extended to include the methodology evolution, two classes must be replaced by their intersection: The instance of the new class contains both previous classes. This is not possible in UML, since every instance can be only the instance of a class and not the instance of two classes at the same time.

It is important to have a machine-readable description of the model. In MOF approach we use XML metadata interchange (XMI) as a model representation language (but we are free to use any XML description). XMI is an OMG standard and there are different formats according to the graphic editors that produce it. A model description must be understandable in an easy and univocal way by the software agent and preferably should be a W3C standard.

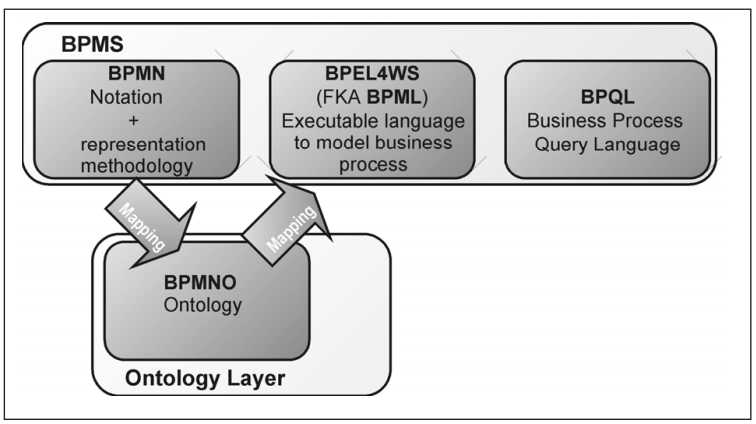
Finally, classes, attributes, and relationships are insufficient to express all the methodology primitives and so, in the MOF approach there is an external language, OCL, to describe the methodology constraints.

THE ONTOLOGY LAYER: ONTOLOGY REPRESENTATION OF BPMN NOTATION

It is clear that to define all business process design details it is a hard task and cooperation between business and IT experts is a must. These two types of users must work on the same project and each of them must add the right detail to the design based on their point of view. To insert the process design in the overall IS architecture, that is, to make explicit and tangible the knowledge embedded in the mind of IT and business experts, it is important to represent in a machine-readable format the overall business process design with all details.

The project of BPMI is to define standards in order to cover the overall chain starting from the business process design (through BPMN notation) to a business process execution language, and finally, the efforts will be focused on a business process query language that allows us to reach, through the right questions, information about processes. The choice of BPMN notation to design business process does not tie to a particular machine-readable format because, although there are big efforts in this direction, there is not a standard machine-readable format to represent

Figure 4. The ontology layer



business processes. Starting from these problems in our research work, our idea (Figure 4) is to add an ontology layer. The choice of ontology (in our approach we adopt Semantic Web technologies different from the Semantic Web idea) helps us to provide, immediately, a process representation in a machine-readable format and, thanks to its flexibility, ontology will help, when necessary, to translate in any formal language the model obtained when this formal language will be defined and will be standard.

BPMN ONTOLOGICAL METAMODEL: OUR APPROACH TO SOLVE MOF PROBLEMS

In order to solve the MOF problems highlighted, we look to other languages different from MOF. In our research work we adopt an innovative language more expressive than MOF: we choose OWL.

The architecture that we adopt in our work is the MOF 4-level architecture but the language in the M3 level is OWL, instead of MOF language thus, the metamodel in M2 level is made up of ontological classes and ontological properties linked together. Finally in the M1 level we obtain the model through instantiation of classes and property previously defined. The M0 level represents, also in our approach, the data of a specific model.

Ontology and OWL as metamodel definition languages help us to obtain several advantages such as:

- **Metamodel semantic:** OWL allows us to define a semantic to what we represent through classes and properties that allow us to express characteristics of classes.
- **Semantic relationship:** OWL and ontology allow us to define ad hoc relationships different from UML where there are only two types of relationships: aggregation and association.
- **Standard description of the model:** By using OWL it is possible to obtain a machine-readable description of the model that a software agent may read in univocal way. OWL is supported by W3C differently from other formats such as XMI (XMI is the export format starting from UML model). XMI is an OMG standard (and not W3C) but there are different formats based on the graphical editor that allow us to define the model.
- **Graphical representation:** Ontological languages are based on text and not on a specific notation so it is possible to provide to the metamodel and to the model a specific graphical representation based on a methodology and not on a general representation.

Our research contribution is oriented to use ontology in a different way from the Semantic Web technologies, which is the traditional one. Following the 4-layer architecture proposed by MOF, we focus on the M3 and M2 levels. In the M3 level (metamodel level) we define all BPMN primitives through classes and properties. Classes and properties are, in some cases, not sufficient to express all BPMN primitives so we also adopt instances of some classes and properties to define the metamodel. The M2 level is made up only by instances of classes and properties that have already been defined. Classes and properties (the metamodel) are the guidelines for the design in order to define the model, that is, to insert the instance of the model.

We develop an ontological metamodel where classes and properties are defined in order to express all BPMN primitives. In our ontological BPMN metamodel we define not only the main primitives but also properties of each of them.

In our approach we develop BPMN metamodel following different steps:

- Analysis of BPMN specification in order to extract the main concept: each concept is defined as ontological class.
- Analysis of BPMN in order to extract details of each concept defined in the previous step: each concept is modeled as ontological subclasses tied to the main classes.
- Analysis of BPMN in order to extract concepts that support the concept defined in the previous steps. Each concept is defined as ontological class.
- Analysis of BPMN in order to extract properties that allow us to provide a semantic to concepts previously defined. It is important to define both Object properties that allow us to link together concept and Data Type properties, that is, simple type.
- Analysis of BPMN in order to reach some concept that is not modeled by classes and properties but as an instance of classes and properties.

In the following section we explain in detail the BPMN ontological metamodel.

DEVELOPMENT OF BPMN ONTOLOGICAL METAMODEL

In the development of the BPMN ontological metamodel we follow the BPMN specification and we try to translate in ontological classes the main concepts defined in BPMN specification.

In the BPMN ontological metamodel we define two main types of classes: Concrete and Abstract classes. Concrete classes are classes that may contain instances when we define a specific model starting from the metamodel. Abstract classes are used only to define BPMN concepts but these classes cannot contain instances of a specific model. Each Abstract class has at least one Concrete class where it is possible to define instances.

In the BPMN metamodel we define both the four different groups of primitives defined in BPMN specification and two other concepts: the concept of the business process diagram and the concept of process.

- **Business process diagram.** It is a Concrete class; it contains general information about design such as author, creation date, and so on. Following the BPMN specification a business process diagram is made up of several Pools.
- **Process.** This concept has been defined to contain the process design that is all the BPMN elements that allow us to define different steps in the process execution design. Process is a Concrete ontological class and has three ontological subclasses of type "Specialization."[1]**AbstractProcess, PrivateProcess, and CollaborativeProcess**. in order to meet the BPMN definition of Process.
- **Swimlane:** This concept has been defined in order to make a generalization of Pool and Lane. *Pool* and *Lane* are concrete subclasses (of type "specialization") of the abstract class Swimlane. The concept of Pool, following the BPMN definition, allows us to define an actor (a person or/and a machine) of the process. A Pool may contain a Lane, Flow Object (defined below) or nothing. The ontological class Lane meets the concept of Lane defined by BPMN and is defined in order to allow the definition of a Lane within a Pool.
- **FlowObject.** With regards to following the BPMN specification, the ontological Abstract class FlowObject is defined as a superclass that contains three subclasses: Activity, Events, and Gateway. The abstract class FlowObject is linked to the concrete classes Activity, Events and Gateway with a "Specialization" relationship. Both Activity, Task, and Event have a subclass that allows us

to define the specific characteristics defined in BPMN specification. As an example to define three different type of Event (Start, Intermediate, End) we define three different subclasses of the class Event.

- **Artifact.** With regards to following the BPMN specification, the ontological class Artifact allows us to define information not tied to the process flow. Ontological class Artifact (an Abstract class) contains three Concrete subclasses Annotation, Data Object, and Group.
- **ConnectingObject.** With regards to following BPMN notation, Connecting Object is defined as a superclass that contains three different subclasses SequenceFlow, MessageFlow, and Association Flow.

The abstract class GenericBusinessProcessObject is the generalization of the last four concrete classes in the bullet item (SwimLane, FlowObject, Artifact and ConncetingObject). In the GenericBusinessProcessObject class we define the datatype property shared by these four classes. The datatype property are:

- **Categories.** In BPMN specification it has documentation purpose; in the metamodel it is a datatype property of type "text."
- **Documentation.** As categories, it is a datatype property of type "text."
- **Name.** It is a text data type property that allows us to define a unique name in the business process diagram for each Generic Business Process Object.

Properties defined in the Abstract classes cannot contain instances but, thanks to the class-subclass the relationship property will be inherited by subclasses until the subclasses will be concrete.

At this point the main concepts of BPMN are represented as ontological classes. In order to link together the main concepts we define the Object Property in the proper classes.

The use of Object Property in the BPMN ontology is a little different from the traditional Semantic Web. An example is useful in understanding this interesting aspect. Each process may be composed by different GenericBusinessProcessObject, and it is not a must to define in each process all the GenericBusinessProcessObject defined in the BPMN specification. If each GenericBusinessProcessObject is defined only by its name a solution may be to define in the class Process several properties (datatype properties) each of one of the generic business process. The generic business process is a more complicated concept: It has several subclasses and each of them has its own properties. To solve this problem in the metamodel that we developed, we adopt an Object Property "hasGenericBusinessProcessObject," which has the class Process as domain and the class GenericBusinessProcessObject as range. The OWL code is in Figure 5.

Figure 5. OWL code of hasGenericBusinessProcessDiagramGraphical Object

```
<owl:InverseFunctionalProperty
rdf: ID="hasGenericBusinessProcessObject">
    <rdfs:range rdf:resource="#BusinessProcessObject">
    <rdfs:domain rdf:resource="#Process"/>
    <rdf:type
  rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
     </owl:InverseFunctionalProperty>
```

In this way it is possible, when defining the model starting from the metamodel, to define several instances of the property "hasGenericBusinessProcessObject" each of them define a specific business process object with its own properties. Starting from this example, we define, in the same way, the property "hasSwimlane." This property has the ontological class BusinessProcessDiagram as domain and the ontological class Swimlane as range. Finally, we define the property "isMadeOfLane" to state that each Pool (class Pool is the domain of this property) may contain one or more Lane (range of property).

Starting from previous consideration the core classes and main relationship of the ontology metamodel are represented in Figure 6.

Some special cases have been faced during the development of the BPMN ontology metamodel.

A Pool, following BPMN specification, may contain both a Lane and a GenericBusinessProcessObject (different from Swimlane) (Figure7). The problem is how to model this concept: make two different Object Properties or the same? The best solution, following the ontology idea, is to provide the same Object Property because the semantics of the relationship are the same. We define the Object Property "belongsToPool" with only one class as range (the ontological class Pool) and the domain as the union of the other classes: Flow Object, Artifact, and Connecting Object. In this way the same property, depending on the context, is used to express both the fact that a Lane belongs to Pool and to lay Flow Object, Artifact, and Connecting Object to the specific Pool.

In the same way, the Object Property "belongsToLane" (Figure 8) is used both to model the fact that one Lane can contain another Lane and to define which Flow Object and/or Artifact are defined in the same Lane.

Additional Classes

In order to cover all the BPMN complexity, during the BPMN metamodel development, we define concepts modeled as ontological classes, not clearly defined in the BPMN specification. As an example we consider the class Trigger. We observe that a trigger is the mechanism that allows an event to start, so a trigger is what allows the events to start. BPMN specification defines some properties for a trigger; for example, if a trigger is of type Timer, it has the property timeCycle that defines the duration of the event and timeDate that define when the event starts. We link the Ontological class Trigger with the Event by the property "hasTrigger." The class Trigger is made up of several subclasses each of them, following BPMN specification, expressing a special type of trigger (Figure 9).

Finally, to define all the BPMN properties of each BPMN primitives we define, where appropriate, ontological properties to meet the BPMN specification.

From the BPMN Metamodel to the Ontological Business Process Model

Starting from the metamodel, previously defined, it is possible to define a business process model defining instances of ontological classes and properties (we talk about concrete ontological classes). Suppose that we want to define a simple business process made up of one Pool and of a star event, one task and the end event. The task is linked to the start and end event with sequence flow.

We define an instance of the class Process and we define the instances of all (of some of them) properties defined by BPMN specification and in the metamodel. Following the property "isDefinedInPool" we define a Pool (and all its properties). Following the property "hasGenericBusinessProcessDiagramObject" (starting from the class process) we define the Start Event, the task, the End Event and two sequence flow: one

Figure 6. Core classes and relationship

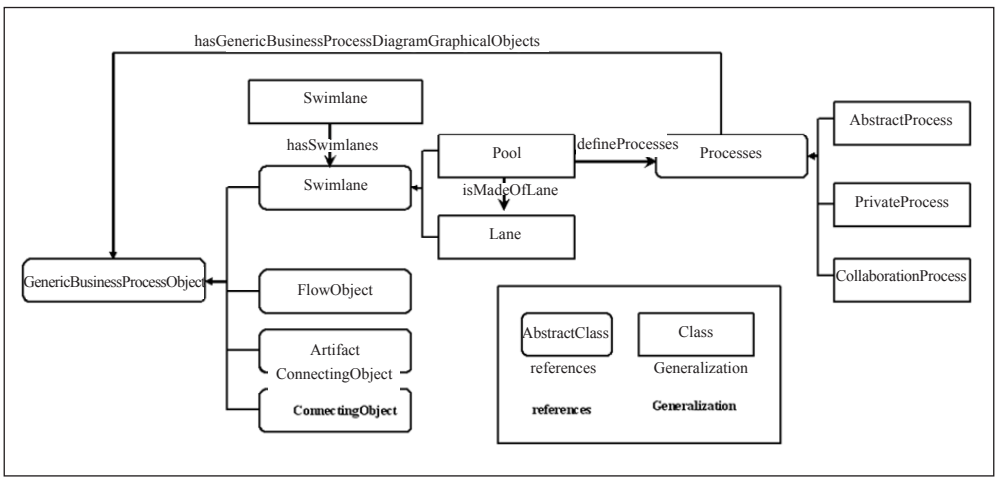


Figure 7. Ontological property belongsToLane and belongsToPool

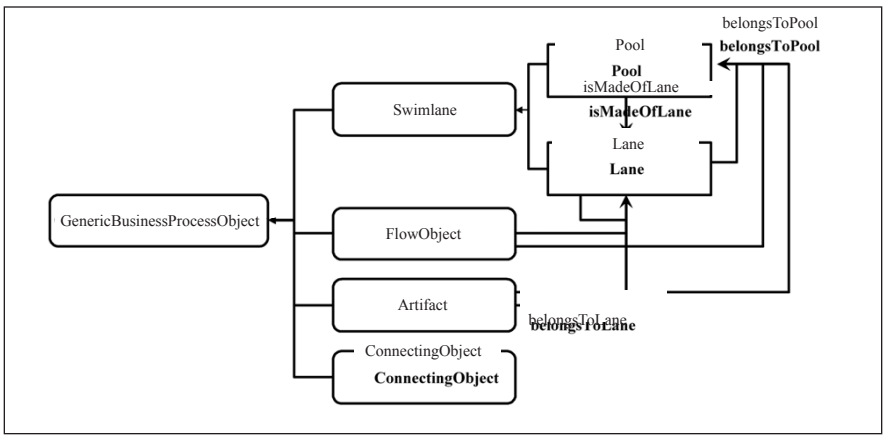


Figure 8. OWL code of the "belongsToPool" properties

```
<owl: ObjectProperty rdf:ID="belongsToLane">
   <rdfs:range rdf:resource="#Lane"/>
   <rdfs:domain>
    <owl:Class>
       <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#FlowObject"/>
          <owl:Class rdf:about="#Lane"/>
          <owl:Class rdf:about="#Artifact"/>
       </owl:unionOf>
      </owl:Class>
   </rdfs:domain>
```

has the start event as source and the task as target and another has the task as source and the end event as target.

Obviously, the BPMN ontological metamodel development is supported by our editor where it is hidden the BPMN complexity and the final design (with or without all details) may be saved in OWL format. The process representation so obtained follows the metamodel and it is an instance of it.

FUTURE TRENDS

Starting from an ontological definition of the BPMN metamodel and thus from an ontological definition of the model, our next step is oriented to two different directions.

One way is to understand the possible key performance indicator and thus to design and implement a tool able to make the *simulation* of the process in order to provide to the manager the possibility to understand possible management and/or design errors and to correct them immediately. The ontology representation of the process may help in this work thanks to the possibility to associate rules to the ontology and thus to think on a reasoning tool.

Starting from the idea to define a light framework to manage processes and to help in the design and implementation of Web application based on a process design, we look to the *guidelines* that support the steps from process design to Web application design. Guidelines will be, as soon as possible, a *Web application design methodology* that supports the process definition: This design methodology will cover the gap in the traditional Web application design methodology based on a "user experience" paradigm but not focused on a process.

CONCLUSION

Starting from the necessity to introduce the process management in the overall IS architecture, our work focuses on the description of business process in a formal way aimed to automate the business process management. The focus is first on the notation to adopt: We think that the notation must be complete and expressive with the goal to cover the traditional semantic gap between business and IT experts. Our choice is the BPMN notation because BPMN goal is nearer to our goal. We focus also on the formal description of the business process in a machine–readable language. Observing that to represent the process we need a metamodel that is a guideline in the model definition, the focus on the 4-layer MOF metamodel is very helpful for our purpose but we highlight that the UML language used is poor for several reasons. So we introduce an innovative way to think on ontology different from the traditional Semantic Web use. OWL provides us several advantages both in the description of the metamodel and in the description of the model. OWL and Semantic Web technologies will help us in our future works.

REFERENCES

Berners-Lee, T., Hendler, J., & Lassila, O. (2001, May). The Semantic Web. *Scientific American, 284*(5), 34-43.

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition, 5*, 199-220.

Hammer, M. (1990). *Reengineering work: Don't automate, obliterate. Harward Business Review, 68,* 104-112.

Miers, D., Harmon, P., & Hall, C. (2006). *The 2006 BPM suites report Realise 2.0.* Retrieved

September 30, 2006, from http://www.bptrends.com/reports_toc_01.cfm

Retrieved March 15, 2005, from http://www.bptrends.com

Object Management Group (OMG). (1997, September). *Object constraint language specification* (Version 1.1).

Object Management Group (OMG). (2001, September). *Unified modeling language specification* (Version 1.4).

Object Management Group (OMG). (2003, September 8). *UML 2.0 superstructure specification* (Version 2.0).

White, S. A. (2004, May 3). *Business process modeling notation (BPMN)* (Version 1.0). Retrieved May, 2004, from http://www.bpmn.org

World Wide Web Consortium (W3C). (2004a, February 10). OWL *Web ontology language reference*.

World Wide Web Consortium (W3C) (2004b, February 10). *RDF vocabulary description language 1.0: RDF Schema*.

World Wide Web Consortium (W3C) (2004c, February 10). *RDF/XML syntax specification*.

ENDNOTE

[1] Specialization subclass is also called IS-A, and it allows us to connect a general concept with a more specific concept. In our example an AbstractProcess IS-A Process.

This work was previously published in Semantic Web Technologies and E-Business: Toward the Integrated Virtual Organization and Business Process Automation, edited by A. Salam and J. Stevens, pp. 77-100, copyright 2007 by IGI Publishing, formerly known as Idea Group Publishing (an imprint of IGI Global).