Towards a Methodology for Representing and Classifying Business Processes

Conference Paper *in* Lecture Notes in Business Information Processing · May 2009

DOI: 10.1007/978-3-642-01187-0_16 · Source: DBLP

CITATIONS
11

READS
184

5 authors, including:

Hafedh Mili
Université du Québec à Montréal
206 PUBLICATIONS **4,698** CITATIONS
SEE PROFILE

Abderrahmane Leshob
Université du Québec à Montréal
39 PUBLICATIONS **239** CITATIONS
SEE PROFILE

Ghislain Lévesque
Université du Québec à Montréal
20 PUBLICATIONS **196** CITATIONS
SEE PROFILE

Ghizlane El-Boussaidi
École de Technologie Supérieure
80 PUBLICATIONS **890** CITATIONS
SEE PROFILE

Towards a Methodology for Representing and Classifying Business Processes

Hafedh Mili, Abderrahmane Leshob, Eric Lefebvre, Ghislain Lévesque, and Ghizlane El-Boussaidi

LATECE Laboratory, Université du Québec à Montréal,
B.P 8888, succursale Centre-Ville, Montréal (Québec) H3C 3P8, Canada
first.last@uqam.ca, eafrlefebvre@videotron.ca

Abstract. Organizations build information systems to support their business processes. Some of these business processes are industry or organization-specific, but most are common to many industries and are used, modulo a few modifications, in different contexts. A precise modeling of such processes would seem to be a necessary prerequisite for building information systems that are aligned with the business objectives of the organization and that fulfill the functional requirements of its users. Yet, there are few tools, conceptual or otherwise, that enable organizations to model their business processes precisely and efficiently, and fewer tools still, to map such process models to the software components that are needed to support them. Our work deals with the problem of building tools to model business processes precisely, and to help map such models to software models. In this paper, we describe a representation and classification of business processes that supports the specification of organization-specific processes by, 1) navigating a repository of generic business processes, and 2) automatically generating new process variants to accommodate the specifics of the organization. We present the principles underlying our approach, and describe the state of an ongoing implementation.

1 Introduction

Organizations build information systems to support their *business processes*. One is tempted to infer that two organizations that employ the same business processes should be able to use (or reuse) the same IT infrastructure. At a very high-level of abstraction, this may very well be the case (e.g. BIAIT [1]). However, reuse at the IT strategic or enterprise architecture level does not translate into reuse at the more concrete software artifact level (models, at all levels, and code), where most of the development and maintenance resources are spent. There are many reasons for this. First, there is a wide variety of business processes for doing anything, from purchasing, to inventory management, to billing, or hiring, which may coincide on the fundamentals, but differ in the detail. Further, for any given business process, there are different levels of IT support, ranging from a simple recording of the activities of an essentially human business process, to performing the clerical steps of the process, and recording the others, to a full process automation. The BIAIT methodology, for example, recognizes some of these variations and encodes them in half a dozen binary decisions.

G. Babin, P. Kropf, and M. Weiss (Eds.): MCETECH 2009, LNBIP 26, pp. 196–211, 2009.

However, this comes far short of capturing all of the possible variations in the business processes, and in the level of support provided by the IT system.

Add to the above variations the great variety of target domains, or industries, such as banking, insurance, manufacturing, pharmaceuticals, etc. But how different are the business processes across domains, really? The process of *selling* computers is similar to the process of *selling* cars, much more so than to the process of *manufacturing* computers. In fact, because cars and computers share many characteristics—both are tangible, somewhat high-priced, manufactured, configurable products—the processes are nearly identical. What distinguishes the two is the domain vocabulary (computers vs. cars, processors vs. engines, etc.). A good fraction, if not the majority, of the business processes of an organization do not depend on the industry within which it operates. However, the *analysis models* of the information systems that support them will be domain-specific.

Our long term objective is to characterize the transformation from business process to software with enough precision to be able to instrument it with tools that help business and system analysts, working together, to generate a first sketch of the target software system based on a precise model of the business processes that it supports. Traditional approaches to this problem are catalogue-based: we build a catalog of software components that are somehow *indexed* by the *elementary* business processes that they support. To differing degrees, this is the approach taken by the IBM San Francisco initiative and by SAP through its 'blueprint'. The San Francisco initiative suffers from the relatively low granularity of the components, creating a significant semantic gap between the business process level, and the software component level. Further, lots of glue is needed to assemble the low-granularity components, diminishing the reuse effectiveness of the approach. For the case of SAP, the business processes are at the right level of granularity, but the mapping from business process to software is embodied in proprietary tools, and there is still quite a bit of customization required.

Our approach to this problem consists of precisely characterizing and codifying the three sources of variability that we mentioned:

1. Process variability: accommodating differences in business processes to accomplish the same business objective.
2. Domain variability: accommodating differences between application domains.
3. Automation variability: accounting for the fact that different information systems will offer different levels of automation for the same processes.

Our approach to handling the first two sources of variability relies on a combination of, a) a catalog of generic business processes, b) a representation system for such business processes that supports a number of specialization operators enabling us to generate on-the-fly specializations, and c) a mapping procedure that enables us to *instantiate* a generic business process for a particular domain. The difference between our approach and other catalogue-based approaches (e.g. [8]) is that our catalogue does not need to be exhaustive: we can generate new process variants (specializations) on-the-fly, as opposed to having to encode them manually.

In the next section, we describe the process modeling language that we will be using. Section 3 talks about how we handle process variability and domain variability. In particular, we discuss our approach to generating and representing process specializations.

Section 4 describes the design of a toolset for the representation and classification of business processes. Section 5 describes the current implementation. We conclude in section 6 by highlighting directions for future work.

2 Process Modeling

Curtis defined a process as a partially ordered set of *tasks* or *steps* undertaken towards a specific goal [3]. Hammer and Champy define *business processes* as a set of *activities* that, together, produce a result of value to the customer [5]. The workflow management coalition defines business processes as "a set of one or more linked procedures or activities which collectively realize a business objective or policy goal, normally within the context of an organizational structure defining functional roles and relationships." [9,][10]. We adopt the definition embodied in the meta-model of Fig. 1. The **activities** of a business process are performed by **actors** playing particular **roles**, consuming some **resources** and producing others. Activities may be triggered by **events** and may, in turn, generate events of their own. The **activities** of a process may be linked through **resource dependencies** (producer-consumer dependencies) or **control dependencies** (one activity triggering another). The **actors** operate within the context of **organizational** boundaries. Organizations perform specific business **functions**. Roles can support **functions**. We will refer again to this meta-model when we present our classification procedure.

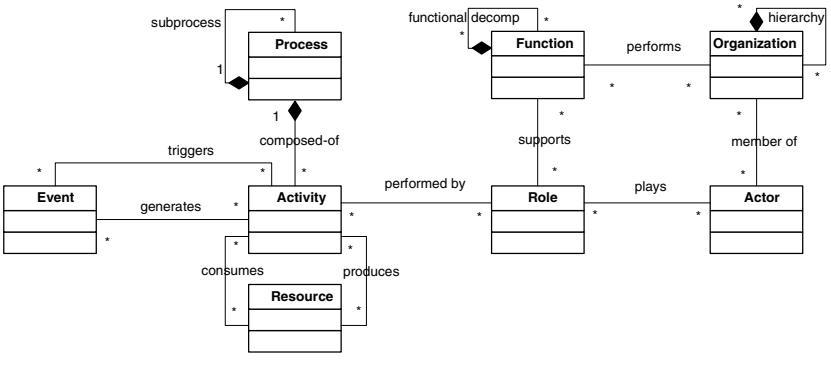

Fig. 1. A first-cut business process metamodel

As our process metamodel suggests, there are a number of things to represent about a process. Curtis argued that there are four distinct views [3]: (1) *The functional view* presents the functional dependencies between the process elements, such as producer-consumer dependencies. (2) *The dynamic view* provides sequencing and control information about the process, i.e. when certain activities are performed, and how. (3) *The informational view* includes the description of the entities that are produced, consumed or otherwise manipulated by the process. These entities include pure data, artifacts, products, etc. (4) *The organizational view* describes *who* performs each task or function, and *where* in the organization (functionally and physically).

Most object-oriented modeling notations cover the first three views. What is new, from an ontological point of view, is the *organizational* view, which includes a description of the *participants* in the process as well as a description of the physical (location) and organizational context within which this process is conducted. Further, whereas the informational view in object-oriented modeling represents only data entities, the informational view of business process modeling may represent tangible resources and artifacts that are used and produced by processes.

We studied a dozen or so process modeling languages that originated from a variety of scientific traditions (see e.g. [11]). Because we want to *ultimately* map our process models to information system object models, we used UML 2 as a basis, and used its extension mechanisms to introduce the organizational view.

3 An Approach to Business Process Classification

3.1 Principles

For the purposes of illustration, we will use the example of an ordering process. Ordering starts by first filling out a request for a product which then goes through a budgetary approval process. If it is approved, it goes to purchasing, who will identify suppliers for the product. Once a supplier is found, then a purchase order to *that supplier* is created and sent. When the product is received, it is sent to the requester. Then payment is made. Fig. 2 shows a simplified functional view of the process.

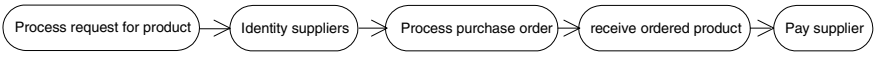

Fig. 2. The functional view of a basic ordering process

This process is independent from the application domain, and at this level of abstraction, it can be used to order pencils or computers or airplanes. There are many variations of this process, notwithstanding the target application domain. For example, for on-line purchases, we typically pay before "receiving" the product, because of the anonymity (and impunity) of the internet. Second, if the buyer has a running contract with a supplier, then they do not look for suppliers each time: they order directly from that supplier. Third if the requester is also the decision maker, they do not need to ask for approval: they can just go ahead and order the product.

These are all variations that do not depend on the target industry or application domain. Naturally, we expect the software applications that support the purchasing process to exhibit similar variations. This raises two questions:

1. Is there a way to organize existing business processes in a specialization hierarchy that makes it easy for users to navigate to find the business process that best fits their organization,
2. Is there a way to *generate* some of these specializations on-the-fly based on some catalog of elementary specializations,

We discuss each question briefly. Subsection 3.2 presents our approach.

There have been many initiatives aimed at cataloguing generic business processes, each proposing classifications of their own, including the MIT process handbook, the IBM San Francisco project, and various analysis pattern catalogues. These classifications are for the most part high-level, and refer to broad functional areas such as *production*, *logistics*, *support*, or *planning* (e.g. [2],[7]). These classifications are also *descriptive* in the sense that they are based on external properties of the process (*meta-data*) as opposed to *structural* classifications, which are inherent in the structure of the models—and can be computed from it. The MIT process handbook uses a descriptive classification, in addition to a question-based classification discussed in the next section. Descriptive classifications require little automation, and are easy to implement. They are, however, labor intensive. Structural classifications would help us answer the second question, i.e. generate on the fly process specializations based on a catalogue of generic processes and a catalogue of elementary specializations.

Fig. 3 shows a simplified model from the *informational view* of the ordering process. As mentioned earlier, the ordering process would depend on the existence of a *contract* between the buyer and the appropriate supplier regarding the terms of purchase (price, delivery delays, defect return policy, etc.), which will obviate the need for searching for a supplier. Second, the reception of the product will depend on whether the product is a tangible product (a chair, a computer, a book), or a non-tangible product, or *service*, such as internet access, phone service, etc.

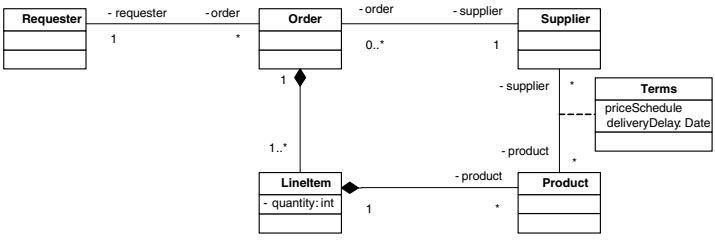

Fig. 3. Basic (partial) informational view for the ordering process

Figure 4 shows a new object model (informational view) that accommodates both of these changes. The new model is similar to the original one, with two differences (noted in grey boxes): 1) we added a class (**Contract**) and two associations between the new class and existing ones, and 2) we specialized an existing class (**Product**) into two subclasses (**TangibleProduct** and **Service**). This simple example raises a number of points that we discuss below.

Let us first start with the specialization of **Product** into **TangibleProduct** and **Service**. We, in the object world, are familiar with these kinds of specializations. In framework speak, these are called hotspots, which are well-defined points of extension in object models using well-defined extension mechanisms; in this case, sub-classing. Sub-classing or sub-typing only covers the simplest cases. With the contract example, we are adding a class and two associations, and there is no way of guessing that the addition actually *specializes* the process. Third, adding a contract between buyer and supplier actually removes one step from the functional view. It also modifies the dynamic view accordingly.

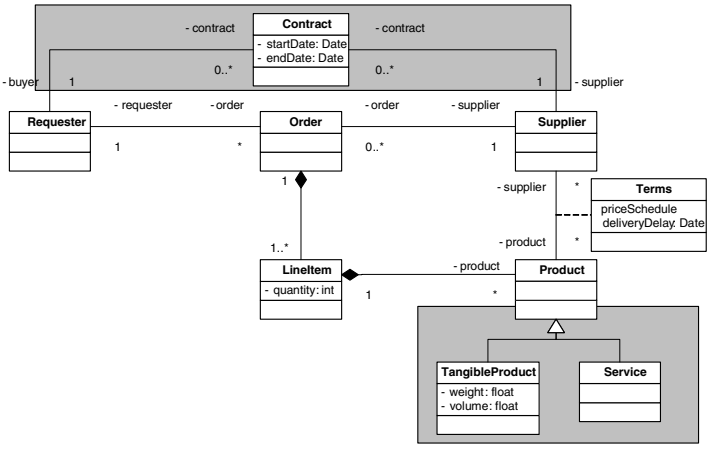

Fig. 4. The informational view for a specialization of the purchasing process

This illustrates a problem first raised by Lubars in 1992 [6]. Objects have been originally sold on the intuitive belief that "small" changes in requirements result in "small" changes in the corresponding program, thanks to inheritance and encapsulation. Lubars showed that this *may* be true for the object model, but not so for the dynamic model, for example: small changes in requirements can result in dramatic changes in the dynamic model [6]. What all this means is the following:

1. what we might intuitively refer to as *process specialization* may have a simple expression in one of the four views, but not necessarily in the others
2. the specialization operators depend on the view, and may not be related to the object-oriented specialization or extension operators
3. one specialization will affect several views differently.

The answer to our second question is then, yes, it might be possible to generate process specializations on the fly using a catalogue of elementary specialization operators, but that catalogue will have to include far more than the typical object-oriented ones (composition, inheritance).

3.2 Classification Using Metamodel Hotspots

Our analysis of the business process classification problem above showed that there are no simple specialization operators that can be applied on any of the views that would yield systematically meaningful specializations. Some previous work has used *questions* to derive specializations of processes. Carlson argued that the purpose of any organization is to offer a product or a service to a client, and hence, an information system that supports the organization would need to manage this "ordering" process [1]. The data and the operations supported by the information systems depend on the business model and on the way the organization works.

Carlson had reduced these variations to the answers to seven questions whose answers (yes/no) determine the kind of process, and thus the information system needed

to support it. We show below a couple of questions, and illustrate their implications on the business process and on the corresponding information system:

- *Does the supplier send an invoice to the customer, or does the customer pay for the product/service cash (or equivalent)*? If the supplier sends an invoice, we have an invoicing process and a payment process with checks, wire transfers and the like. Also, the information system will need to keep information about the customer, their billing address, and their banking information. If the customer pays cash, no records need to be kept of the customer.
- *Are the prices negotiated, i.e. they differ from one customer to another, or are they fixed*? Negotiated prices mean contracts, price schedules per customer, etc.
- *Is the product or service leased to the customer by the supplier, who conserves all property rights, or is property transferred to the customer*? If the product is leased, the organization needs to keep track of the leasee and manage the product or service throughout its lifecycle. This also has major implications on accounting.

Lefebvre used a variation of these questions to help identify *software* component archetypes [2]. Notwithstanding the fact that BIAIT's seven questions may not be orthogonal—they are not—the questions are fairly coarse-grained, and alone cannot capture the level of detail required for the processes to be able to generate the corresponding information system models. The MIT process handbook also used questions to specialize processes [8]. However, the questions are process-specific. Using process-specific questions has the advantage that both the questions and the resulting specialized processes are precise. It has the disadvantage that the classification is ad-hoc and cannot be generalized: whoever specifies a generic business process has to classify and encode all the variations that would make sense, manually. Further, we cannot generate process specializations on the fly.

By going over a number of processes from the MIT process handbook, and the associated questions, we realized that the questions are about the *roles* involved in the process (customer, supplier), the nature of the *resources* produced and consumed by the various *activities* (product, service, tangible product), or the *organization* within which activities are taking place. Thus, we can frame our questions generically in terms of entities and associations in the process metamodel, and then *instantiate* them for specific processes—instances of the process metamodel—to get process-specific questions. Some of these questions are more related to the informational view, while others are related to the organizational view, while yet others are more related to the functional and dynamic view. We reproduce in Fig. 5 the partial business process metamodel where we outlined the model fragments included in each view.

We show below a couple of generic questions, how they impact a process, and see how they are instantiated for a specific business process. The view is shown between parentheses:

- *Can an actor play several roles within the process (organizational)*? when an actor plays several roles within the same process instance, the underlying process is generally simplified. In our purchasing example, we have three roles within the purchasing organization involved in the creation of the purchase order: the requester (end-user), the person responsible for the budget, and the purchasing agent. If the requester and the budget person are the same, we don't need approval.

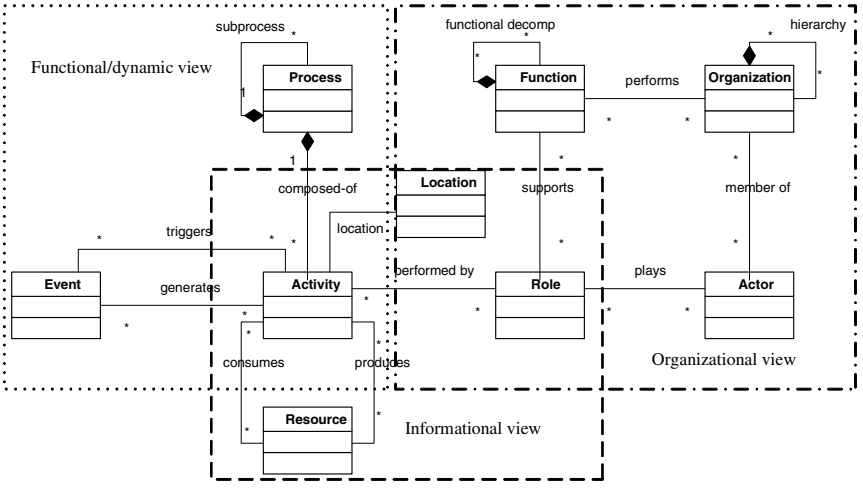

Fig. 5. A partitioning of the business process metamodel

- *Is information about the actors recorded (informational)*? when information about an actor is recorded, the actor is represented internally in the system. Also, activities will be logged in the system. In the purchasing example, this question will be instantiated into several questions, one per actor: a) is information about the requester recorded?, b) is information about the budget approver recorded?, c) is information about the purchasing agent recorded? A yes in each case will have implications on the data and the processing of the application.
- *Does the process execution follow an agreement of some sort (functional)*? if there is an agreement, it means either that process parameters can be obtained from that agreement, or that process execution needs to be validated against the agreement. In the purchasing example, we have the case where a binding agreement exists between the purchaser and a supplier, which simplifies the search for suppliers and initializes some parameters (e.g. price), and the case where no such agreement exists, in which case the order goes to the lowest bidder, for example.

We have identified fifteen (15) questions in all, five organizational, four functional, and six informational. Some of the informational questions have to do with the nature of the resources (tangible vs. non-tangible, perishable or not, degradable through consumption or not, limited quantity or not).

Once we have identified the questions, we have to determine the effect of the answer on the corresponding process models, and more specifically, on each view. Naturally, the questions may impact some views more than others. For each question, we need to develop a set of transformations per view. Some of these transformations consist of removing model fragments that follow a specific pattern, as in removing coordination activities between roles played by the same actor. Others consist of adding model elements (entities, associations, processes) to model fragments that satisfy a specific pattern, in much the same way that we apply analysis or design patterns to existing models. In fact, we are using some of the published analysis and process patterns to this end [2], [4].

4 Designing a Tool for Process Modeling and Specialization

We have started developing a tool set to experiment with the ideas presented in section 3.2. In this section, we present the design principles behind our toolset. In subsection 4.1, we present the concept of operations of our toolset, i.e. how we envision it to be used by process modellers. Subsection 4.2 looks at the representation of process models, specialization questions, and process specializations.

4.1 Concept of Operations

At the core of our toolset is a repository of business processes that process analysts and modelers can browse to find the process that most closely matches the needs of their organizations. In those cases where the best match that is found in the repository is not close enough, process analysts are expected to specialize one of the existing processes of the repository to obtain a process that more closely matches the needs of their organization. Initially, we will have to populate the repository with a set of generic processes that we could take from the MIT process handbook or similar public domain sources. It is expected that usage of our toolset will continually enrich the repository. Eventually, we expect to get to the point where process specialization becomes a rare occurrence, as process modelers will likely find what they need in the repository. Figures 6-a and 6-b illustrate this process.

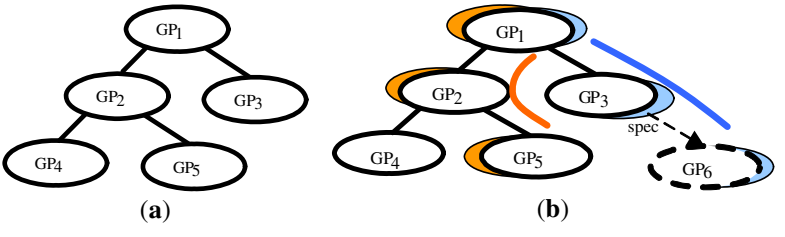

Fig. 6. (**a**) We start with a catalogue of generic business processes, GP_1 through GP_5. (**b**) Modellers find the process they need by using one of the stored ones as-is (e.g. GP_5), or by specializing an existing process (GP_3) using specialization operators to obtain a new process that will be stored in the repository (GP_6).

We envision the proposed system as a pair of applications that access the process repository: a) a web application that enables process modelers to consult the repository in a read-only mode, and b) a desktop application for specializing—and more generally, editing—processes. The main entry point of our system is the web application, which enables process modelers to navigate the process repository (forest). When needed, a process modeler can select a process to specialize, and that takes them to the desktop application. Fig. 7 illustrates this. The desktop application, shown in the left hand side, is identified as an "Eclipse EMF application". Indeed, as explained in section 5, we will use the Eclipse Modeling Framework (EMF) to implement our process metamodel, and use EMF based tools, (graphical) editors, and code generators to build and specialize processes built using that metamodel.

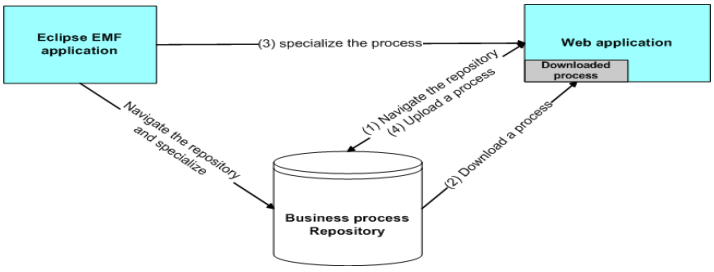

Fig. 7. Proposed specialization system

4.2 Representing Process Models, Questions, and Specializations

Process Models and Process Views. A tool that supports the representation and classification of business processes needs to support each one of Curtis's four views (i.e. *informational*, *functional*, *dynamic*, and *organizational*) Each view consists of a set of model elements appropriate for that particular view. For example, the informational view will contain entities (classes) and associations, whereas the dynamic view will contain activities, events, and transitions. And so forth. As mentioned above, the answer to a question will specialize each one of the views using view-specific specialization operators. A simple metamodel is shown in Fig. 8. We will complete the various pieces in the subsequent paragraphs and in section 5.

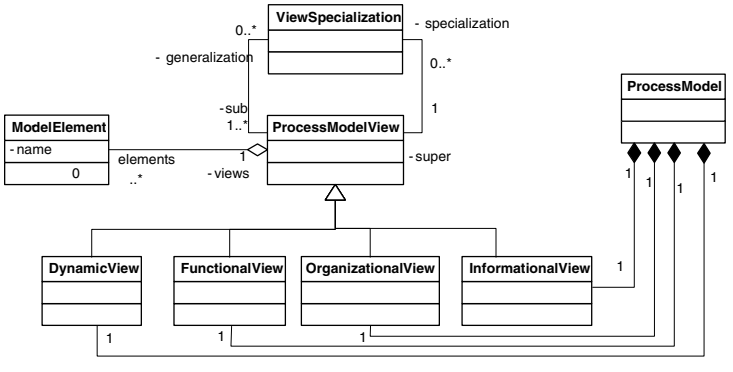

Fig. 8. Business process views

Question model. We can think of generic questions as functions that take a number of parameters and that have a return value. The parameters have *types* which are process metamodel entities, and that correspond to what we called *metamodel hotspots*. The return value is the answer to the question. Typically, questions will have Boolean answers (true/false), but in the general case, we will say that the answer to a question belongs to an "answer type".

Let us start with the example of the generic question "*does the process execution follow an agreement or a contract between* ***Actor*** *and* ***Actor***". Symbolically, we can think of this question as the function:

boolean processExecutionFollowsContractBetween(**Actor** actor_1, **Actor** actor_2);
Here, the type **Actor** refers to the class that is part of the business process metamodel (see Figures 1 and 5).

This function can be instantiated for a specific business process to yield process-specific questions. Consider our ordering process of section 3. That process—its organizational view—includes two actors: a **Requester** and a **Supplier**. The process-specific questions consist of all of the possible invocations of the function above by binding the formal parameters "actor_1" and "actor_2" to actual actors from my ordering process. Because our process has two actors, we get four possible process-specific questions, corresponding to four possible "function invocations":

1) processExecutionFollowsContractBetween(**Requester**, **Requester**)
2) processExecutionFollowsContractBetween(**Requester**, **Supplier**)
3) processExecutionFollowsContractBetween(**Supplier**, **Requester**)
4) processExecutionFollowsContractBetween(**Supplier**, **Supplier**)

The corresponding natural language transcriptions of these functions are:

1) *does the process execution follow an agreement or a contract between* **Requester** *and* **Requester**
2) *does the process execution follow an agreement or a contract between* **Requester** *and* **Supplier**
3) *does the process execution follow an agreement or a contract between* **Supplier** and **Requester**
4) *does the process execution follow an agreement or a contract between* **Supplier** and **Supplier**

If we apply the generic question to a process model that involves three actors, this will result into *nine* different process-specific questions (3 X 3), and so forth.

Clearly, some of these instantiations are not very interesting, and it would have been easy to identify them. For example, in this case, it does not make sense to instantiate the question for a pair of identical actors (<**Requester**,**Resquester**> and <**Supplier**,**Supplier**>). Further, this question is "symmetrical" in the sense that f(x,y) = f(y,x) for a given x and y. Given this information, when we apply the generic question to the ordering process, we should get a single interesting process-specific question, "processExecutionFollowsContractBetween(**Requester**, **Supplier**)", or, in English, "*does the process execution follow an agreement or a contract between* **Requester** *and* **Supplier**". Accordingly, our representation of generic questions includes a *filter* that applies to parameter bindings to eliminate inacceptable or redundant bindings.

Fig. 9 shows our model for representing generic questions and process-specific questions. The **GenericQuestion** class has the attributes name, description, core, and filter. The 'core' attribute is used to represent the natural language template of the question. In this case, it is the string "does the process execution follow an agreement or a contract between {0} and {1}", where {0} and {1} refer to the positional parameters 0 and 1. The 'filter' attribute represents the instantiation filter as illustrated in the above example. The parameters of a **GenericQuestion** are represented by the association class **Parameter** which indicates the parameter name, position (0, 1, etc.). The type of the parameter is represented by the end class **ModelElementType**, which stands for process metamodel entities. The lower part of the model shows how process

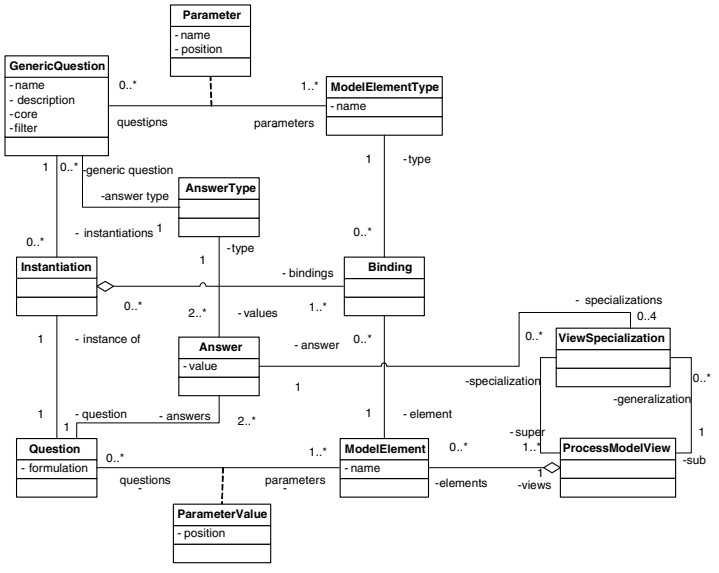

Fig. 9. Question model

specific questions (**Question**) are instantiated from **GenericQuestion** by binding (class **Binding**) formal parameters to actual process model elements. A given **Question** can have two (yes/no) or more answers (**Answer**), and each answer can potentially result into a view specialization[1].

The specialization model. As we have shown in Fig. 9, a specific answer (e.g. "yes" or "no") to a process-specific question, can yield a specialization of one or more of the process model views. We think of these specializations as the application of *view-type specific, generic transformations to process model views*. These transformations are *view-type specific* because each view type (informational, functional, dynamic, and organizational) has its own transformations. They are generic because they are meant to apply to *all* process models—or process model views. Symbolically, with our ordering process and our question about contracts, we can express the specialization (transformation) of the informational view this way:

If (processExecutionFollowsContractBetween(actor_1, actor_2) == true) **Then**
1) Create a new class cls with name '**Contract**' with attributes 'terms', 'startDate', 'endDate', and add it to the informational view;
2) Add an association between cls and actor_1 labeled "binds" with cardinality …. to the informational view;
3) Add an association between cls and actor_2 labeled "binds" with cardinality …. to the informational view;

[1] Not all answers result into a specialization. For example, if the answer to our question "*does the process execution follow an agreement or a contract between* **Requester** *and* **Supplier**"is "no", we keep the same process. Only when the answer is "yes" do we get a more specialized process. The 4 cardinality in the 0..4 is due to the fact that an answer can specialize a process along 0, 1, 2, 3, or 4 views.

We represent the transformations in terms of a set of **TransformationRule**'s. Given a (i.e. one) generic question, an (1) answer, and a view type (1, e.g. informational), we can have *several* transformation rules. In our example here, we expressed the transformation with a single rule, but in the general case, we can have several. Fig. 10 illustrates this. It is actually **the codification of these generic transformations that represents the major conceptual difficulty in our approach**. We think of the **ViewSpecialization** entity as a *trace* of the actual transformation rules as they were applied to a particular view. The "trace" relationship is represented by the many-to-many association between **TransformationRule** and **ViewSpecialization**.

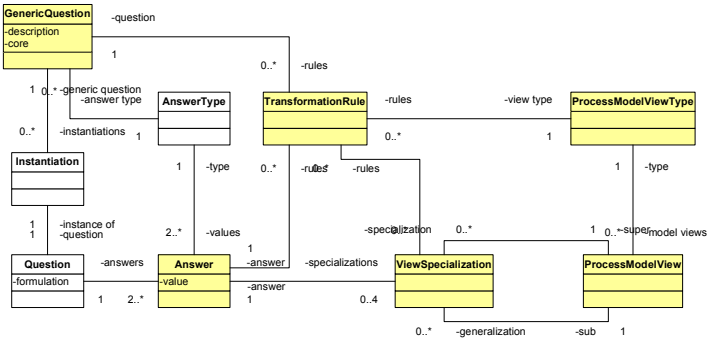

Fig. 10. Process specialization model

5 Implementing Process Modeling and Specialization

In this section, we describe the state of the current implementation of our toolset for modeling and specializing processes. Referring to the concept of operations shown in Fig. 7, so far we worked only on the desktop modeling application; we have not started work on the web application. As hinted at in Fig. 7, the desktop application is based on the Eclipse platform, and uses the Eclipse Modeling Framework (EMF) as the foundation for implementing the process metamodel and the modeling tools.

5.1 Implement the Business Process Metamodel

Figures 1 and 7 showed a (much) simplified business process metamodel. For our modeling tool, we decided to implement a more realistic, preferably standardized, business process metamodel. However, there were many standards to choose from (see [11]). The OMG set-out to define … a standard for unifying these standards, and it came up with the Business Process Definition Metamodel (BPDM, [12]). BPDM is meant to capture the common constructs of the various process modeling languages, while supporting the representation of notation-specific constructs. Consequently, BPDM has a fine conceptual granularity, and we felt that it was far too granular for our purposes. Accordingly, we decided to implement a subset of BPDM.

For the purposes of this paper, we show a *simplified* version of that *subset* in Fig. 11. A Process is defined as a set of **Step**'s sequenced using **ProcessingSuccession**'s.

Steps can correspond to actual work (**WorkDefinitionStep**), e.g. "Fill out purchase order", or control steps (e.g. a fork or a join) or an event step (e.g., a process start or end). Work definition steps consume and produce resources (**ResourceProduct**). **Activity**'ies (elementary work definition steps) are performed by **Actor**'s playing certain roles (**RolePerformer**).

We implemented our business process meta-model using the *Eclipse Modeling Framework*™ (EMF). EMF is a Java modeling framework that implements a core subset of the Meta Object Facility (MOF) API in a package called *ecore*. EMF provides functionality to serialize models implemented with the ecore package into XMI (XML Metadata Interchange) files. So far, we have implemented only the informational and dynamic view.

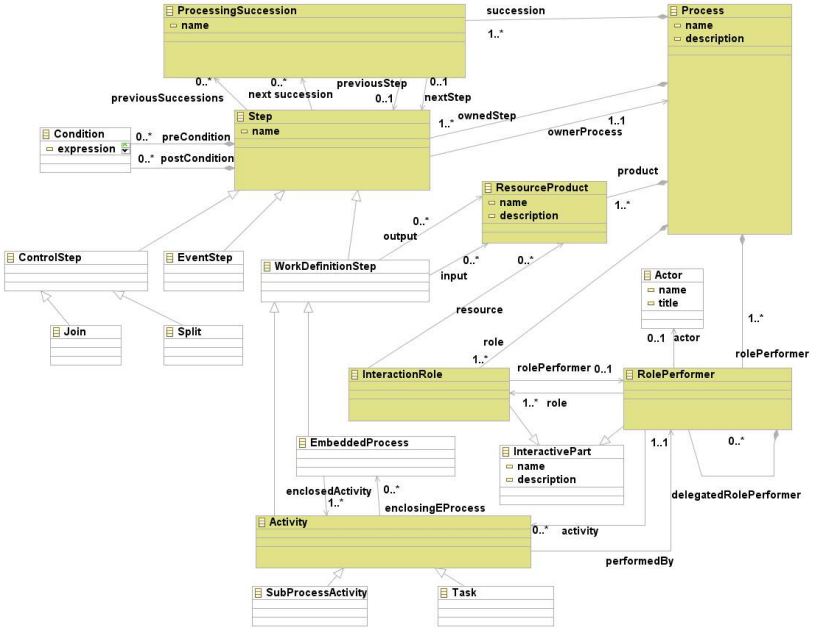

Fig. 11. A simplified view of the implemented business process metamodel

5.2 The Question Model

The generic question metamodel (Fig. 9) was implemented using an XML schema. Hence, generic questions are represented using an XML document. Fig. 12 shows the document that represents the question "*does the process execution follow an agreement between **RolePerformer** and **RolePerformer***". The string that represents the question template is represented by the <core> tag. Note the fillers "{0}" and "{1}", which are like macro variables that stand for the first and second parameters. When this generic question is instantiated for a specific pair of **RolePerformer**'s, the names of these role performers will replace "{0}" and "{1}" in the string. Note also that the instantiation filter is represented by an attribute of the <arguments> tag. The representation of process-specific questions follows a similar structure.

```
<question>
   <name>Binding agreement between role performers</name>
   <description>The process follows an agreement between role performers
   </description>
   <core answerType="java.lang.Boolean" >
         Does the process execution follow an agreement between {0} and {1}?
   </core>
   <arguments filter ="!reflexive, symmetric">
         <argument>RolePerformer</argument >
         <argument>RolePerformer</argument >
   </arguments >
</question>
```

Fig. 12. The representation of generic questions

5.3 Handling Specialization

Recall that process specialization is performed by applying transformations to existing model views, based on the answers to specific questions. We showed in the metamodels of Figures 9 and 10 that each <question,answer> pair may specialize from zero (i.e. no specialization) to four views. We also showed in Fig. 10 that the transformation rules are <question, answer, view> specific. When we started implementing model specialization, we quickly realized that, practically, not all process specializations are worth storing in the repository. Indeed, a process modeler might submit an existing process from the repository to a series of questions, and might be interested only in the final process resulting from applying a string of <$Question_i$, $Answer_i$> specializations, for i=1..n. This means that intermediary process models, those resulting from <$Question_1$, $Answer_1$>, <$Question_2$, $Answer_2$>, …, <$Question_{n-1}$, $Answer_{n-1}$> are not stored: only the starting point and the end point are stored. Accordingly, the specialization link between a process and its specializations is not *always* indexed by a single <question,answer>: it can be a *sequence* of <question,answer> pairs, to which we refer as *process-specific question paths*.

With regard to the transformation rules, we used JBOSS DROOLS, an open-source object-rule production system. Space limitations do not allow us to illustrate our rules. But roughly speaking, the condition part of a rule in our context is satisfied if the rule finds a question with answer "TRUE"[2] while the action part performs the corresponding specialization (e.g. for the *ordering with contract* example, the action part creates the **Contract** class and links it to the classes **Requester** and **Supplier**).

6 Discussion

Organizations build information systems to support their business processes. Some of these business processes are industry or organization-specific, but most are common to many industries and are used, modulo a few modifications, in different contexts. A precise modeling of such processes would seem to be a necessary prerequisite for building information systems that are aligned with the business objectives of the organization

[2] The answer type is String, instead of Boolean. String makes it easier to handle all simple answer types (boolean, int, enumerated values, etc.).

and that fulfill the functional requirements of its users. Yet, there are few tools, conceptual or otherwise, that enable organizations to model their business processes precisely and efficiently. In this paper, we proposed a representation and classification of business processes that supports the specification of organization-specific processes by, 1) navigating a repository of generic business processes, and 2) automatically generating new process variants to accommodate the specifics of the organization.

Our process for specializing process models is question-driven. There are innumerable ways of specializing business processes, but like others before us (e.g. [1], [2], [8]), we argued that process variability can be captured in a relatively small number of questions. Unlike the MIT process handbook approach [8], our questions are generic. Unlike the BIAIT approach [1], our questions are precise enough that they can be linked to specific model entities. Finally, unlike Coad et al.'s archetypes [2], our questions handle all process model views. Central to our approach is the assumption that generic specialization operators can be defined that can specialize any process based on the answer to one of the generic questions. We have had some empirical evidence that this might be true, at least for some questions and class of models.

In this paper, we discussed the design principles of a tool set based on these ideas, and presented the current status of an implementation. This implementation showed the plausibility and *computational feasibility* of the approach. The conceptual work of identifying and codifying the transformation rules has just begun. We believe that this line of work will help in understanding the nature of variability and specialization in business processes and, ultimately, in information models.

References

1. Carlson, W.M.: Business Information Analysis and Integration Technique (BIAIT)-the new horizon. Data Base 10(4), 3–9 (1979)
2. Coad, P., Lefebvre, E., Luca, J.: Java Modeling In Color With UML. Prentice-Hall, Englewood Cliffs (1999)
3. Curtis, B., Kellner, M.I., Over, J.: Process modeling. Com. of ACM 35(9), 75–90 (1992)
4. Fowler, M.: Analysis Patterns. Addison-Wesley, Reading (1997)
5. Hammer, M., Champy, J.: Reengineering the Corporation. Harper Business (1993)
6. Lubars, M., et al.: Object-Oriented Analysis for Evolving Systems. In: ICSE, pp. 173–185 (1992)
7. Maiden, N.A., Sutcliffe, A.G.: Exploiting Reusable Specifications Through Analogy. Com. of the ACM 35(4), 55–64 (1992)
8. Malone, T.W., et al.: Tools for inventing organizations. Manag. Sc. 45(3), 425–443 (1999)
9. Workflow Management Coalition. Doc. WfMC-TC-1016-P (1999)
10. Workflow Management Coalition. Doc.WFMC-TC-1011 (1999)
11. Mili, H., Tremblay, G., Bou Jaoude, G., Lefebvre, E., El Abed, L.: Business Process Modeling Languages: Sorting Through the Alphabet Soup. ACM Computing Surveys (2008)
12. Business Process Definition Metamodel. OMG Document bmi/2007-03-01 (2007)