Towards Building Software Systems from the Specification of the Supported Business Processes

Hafedh Mili, Petko Valtchev, Abdel Leshob, Abdel Obaid, and Ghislain Lévesque

Département d'informatique, UQÀM, Montral (Qc), Canada
first.last@uqam.ca

Abstract. Organizations build software systems to support their business processes. There are different levels of support, ranging from simply recording the activities of an essentially manual process, to performing the clerical tasks of the process, leaving decisions to humans, to fully automating the business process, with rule based decision making. We propose a framework that, given a (precisely specified) business process, and a desired level of support by a target information system, enables us to sketch the general infrastructure of the target software. This is not a new problem, and this goal has been pursued by MIS researchers since the late 70's. However, we believe that we now have the conceptual – and technological – maturity to tackle the problem, in terms of modeling standards, and model transformation infrastructures.

1 Introduction

Enterprises 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. 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 potentially innumerable variety of business processes for doing anything (purchasing, inventory management, billing, 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 [1], for example, recognizes some of these variations and encodes them in half a dozen binary decisions. However, this comes far short of capturing all of the possible variations in, a) the business processes themselves, and b) in the level of support provided by the IT system.

Finally, there is the great variety of target domains, or industries, such as banking, insurance, manufacturing, pharmaceuticals, etc. But how different are the business processes across domains? The process of *selling* computers is similar to the process of *selling* cars, much more so than to the process of *manufacturing* computers. 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 – the platform independent models – will be domain-specific.

Our (very) 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 build a catalog of software components that are somehow *indexed* by the *elementary* business

processes that they support, e.g., San Francisco initiative and the SAP 'blueprint'. The former relies on components of relatively low granularity, creating a significant semantic gap with the business process level. Further, lots of glue is needed to assemble those components, hence the low reuse effectiveness of the approach. For the case of SAP, the business processes have the right granularity, but the mapping to software is embodied in proprietary tools and is therefore customization-intensive.

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.

In handling the first two sources of variability we combine, 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. the MIT process handbook [6]) 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 talks about supporting different levels of automation.

2 Process modeling

Alternative definitions exist in the literature for a process, and for a business process, in particular [5, 7]. 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." [4, 3]. We adopt a variation of this definition as embodied in the meta-model ofFig. 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** that can be supported by roles. This meta-model will be referred to in the presentation of our classification procedure.

There are a number of things to represent about a process. Curtis argued that there are four distinct views [5]:

1. *The functional view* presents the functional dependencies between the process elements, such as producer-consumer dependencies.

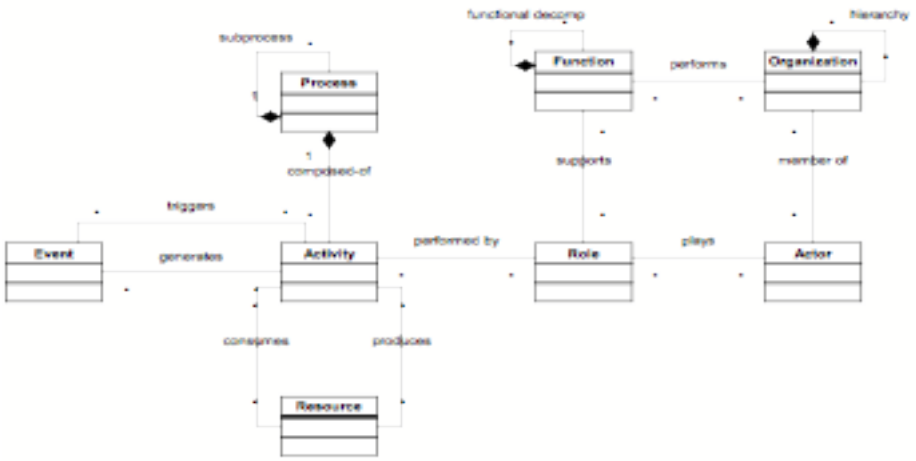

Fig. 1. A first-cut business process meta-model.

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.
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 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 analysis-level class models represent only data entities, the informational view of business processes 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 [10]. Snce 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. This means implementing a metamodel for the organizational view, but also the linkages between the organizational view and the other views, e.g., the dynamic 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 inspected and sent to the requester. Then payment is made. Fig. 2 shows a simplified functional view of the 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. We could have many flavors of this process, regardless of the application domain. For example, for on-line purchases, we usually

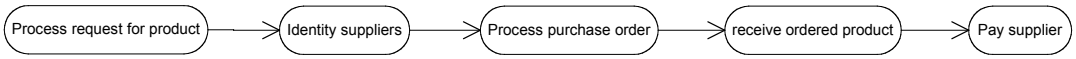

Fig. 2. The functional view of a basic ordering process.

pay before receiving (or activating) the product, because of the anonymity – and impunity – of the internet. Second, if the buyer has a running contract with a supplier, we need not look for suppliers each time: they order directly from the designated supplier. Third if the requester is also the decision maker, they do not need to ask for approval and can simply order the product.

All these variations are domain-independent and represent specializations of the basic business process. 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 users can 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. The next section 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 mostly high-level, and refer to broad functional areas such as *production*, *logistics*, *support*, or *planning* (e.g., [2, 9]). 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. 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. However, they are labour 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.

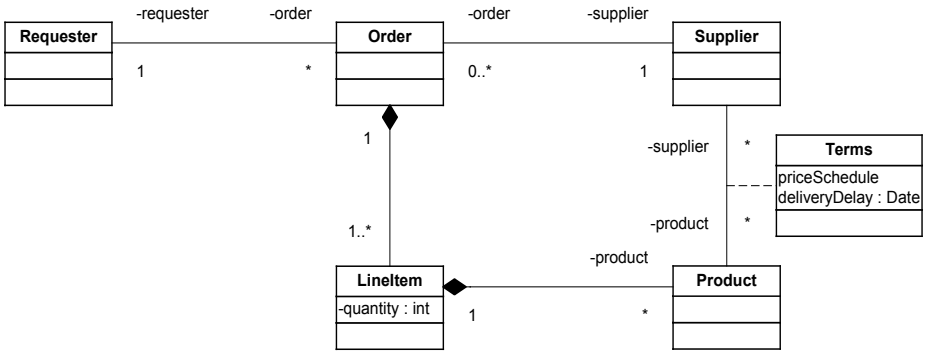

Fig. 3. Basic (partial) informational view for the ordering process.

Fig. 3 shows a simplified *informational* model of the ordering process. As mentioned earlier, the ordering process depends on the existence of a *contract* between the buyer and the appropriate

supplier, which obviates the need for searching for a supplier. Second, the reception of the product depends on whether the product is a tangible product (a chair, a computer), or a non-tangible product, or *service* (e.g., internet access, phone service, etc.)

Fig. 4 shows a new object model (informational view) that accommodates both of these changes. There are two differences between the original model and this one (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 very simple example raises a number of points that we discuss below.

The specialization of **Product** into **TangibleProduct** and **Service** is common and well understood. 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 extensions using well-defined extension mechanisms; in this case, sub-classing. However, sub-classing covers only the simplest cases. The existence of a contract, which *specializes* the business process model, leads to: 1) adding one class and two associations to the informational model, 2) removing one step from the functional view, and 3) modifying the dynamic view accordingly.

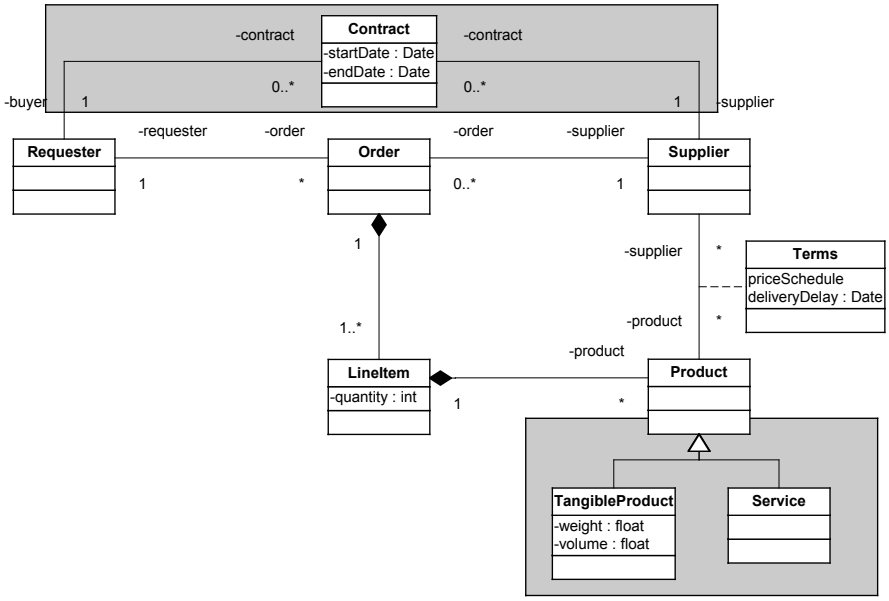

Fig. 4. The informational view for a specialization of the purchasing process.

This simple example suggests 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 question two 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

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 has 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.

Lefebvre used a variation of these questions to help identify *software* component archetypes [2]. However, BIAIT's seven 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 [6]. 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 demand.

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 (e.g. "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 (or phrase) our questions generically about entities and associations in the process metamodel, and then "instantiate" them for specific processes 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 others yet are 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.

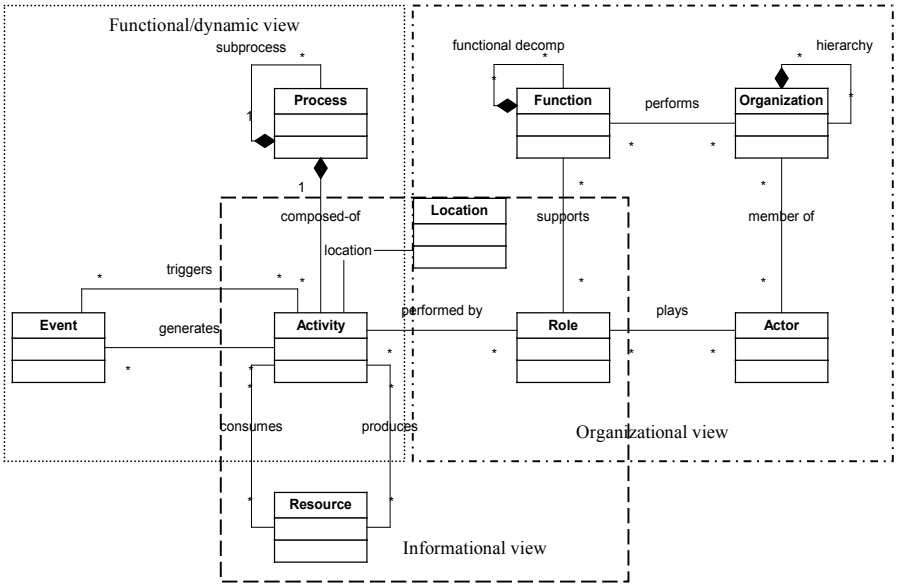

Fig. 5. A partitioning of the business process metamodel.

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, e.g., by removing communication between the two roles. 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.

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 the next section, we discuss the problem of mapping business process models to software models.

4 Mapping Business Process Models to Software Models

In this section, we address the problem of mapping an organization-specific business process to an information system model. We will examine this problem from three different perspectives:

1. The *scope* of the information system. A business process can cover many activities, actors, and organisations, not all of which need to be (or can be) covered by the information system
2. The *purpose* of the information system. Roughly speaking, we can build an information system to *simulate* a business process, or to *automate* it [8], and
3. The level of *automation*. In those cases where we build information systems to automate an information system, what is the degree of automation.

We will first examine the three dimensions in turn, and then discuss the implications of each on the information system. We conclude by sketching a methodology for performing the mapping.

4.1 The scope

The first thing that we need to do is to determine how much of the business process our system will automate. Consider the simple linear process of Fig. 6. In this case, the first activity is to remain manual, as well as all the tasks from $i + 1$ to n.

Accordingly, our system will need to capture information about and potentially perform all of the activities from 2 to i. The amount of information to be captured, and the way to use it, depends on the level of automation (see section 4.3).

In practice, it will often be the case that an information system will not automate a sub-process fully. Take the example of a mortgage loan application process (see Fig. 7). Assume that activities i

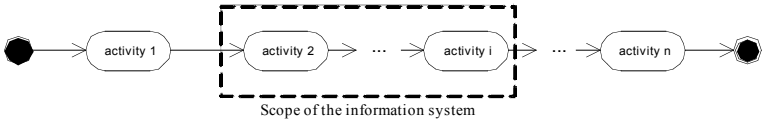

Fig. 6. An information system automates parts of the business process.

to j consist of verifying the eligibility of the borrower (age, residency status, income), and activities m to n deal with property eligibility (location, usage, market value). Before we can assess the property eligibility, we need to conduct an *appraisal* of the property. This is a manual process but we need to record the outcome of that subprocess to assess the eligibility of the property. In this case, the system need not capture detailed information about the intermediary activities ($j + 1$ to $m - 1$); we just need summary information.

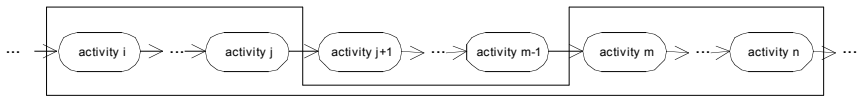

Fig. 7. An information system automates (yet different) parts of the business process.

4.2 The purpose

As mentioned earlier, we can instrument a business process with an information system with two broad goals in mind: simulation, or automation. The mapping from process to software is markedly different. With simulation, the mapping is fairly straightforward: the main entities of the business process (activities, actors, and resources) will find their way into the software as *models* of the real things. Depending on what we want to simulate, different characteristics (attributes) of these entities will be recorded. In this case, the mapping is closer to a *projection*. With automation, the focus is *not* about *reproducing* the manual process – or some aspects thereof – *faithfully*. The purpose is to operate on representations of the state of the world to reflect the changes that happen to it, but the *state transitions* need not have any resemblance to the external processes (physical or abstract) taking place in the business: we only care about the end states.

In practice, most systems would mix the two paradigms. However, we believe that the distinction between simulation and automation needs to be made, and is often the source of lots of confusion with inexperienced modellers who try to transition from a business process model to a software model.

4.3 The level of automation

Consider the following excerpts from the business process meta-model (Fig. 8). What information do we want to capture about an activity. At the very least, we may want to record that the activity has happened, and record the resources that were produced and consumed by the activity. For example, in a sale transaction, we would minimally record that the sale has happened, and we would record the resources consumed – the product that was sold – and the resources that were produced – the payment that was given in exchange of the product. This would be the case for

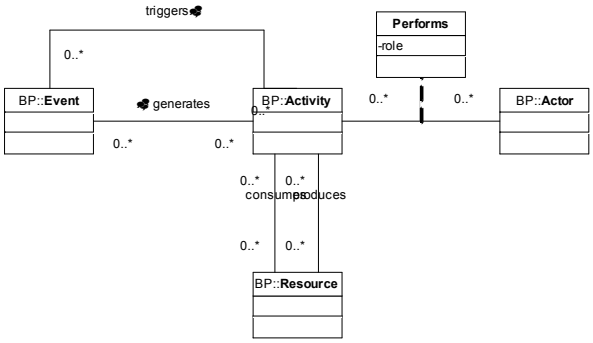

Fig. 8. Excerpts from the business process meta-model.

a simple point of sale (POS) application. The corresponding object model fragment is shown in Fig. 9. We may also want to record information about the actor(s) involved in the activity. In a

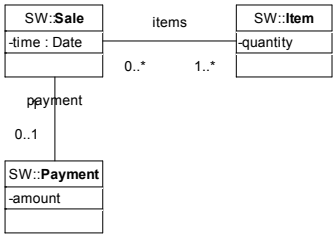

Fig. 9. Sample model fragment for a sales transaction.

sale transaction, there are two actors, the buyer, and the seller. From the perspective of the selling organization, we are always interested in the buyer (e.g. for shipping, customer service, marketing), and often interested in the seller, for tracking, performance, monitoring, etc. If we add information about the two actors, we get the model of Fig. 10. The information about the actors can be extended

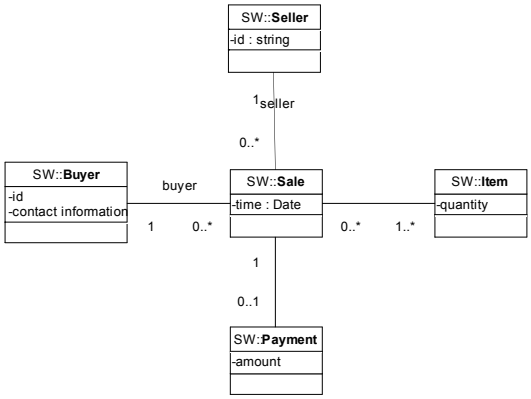

Fig. 10. Sample model fragment for a sales transaction with a representation of actors.

to include the organizational context: both the buyer and the seller belong to some organisation, and that information may be need for accountability or accounting reasons.

An alternative to *recording the occurrence* of an activity is to have the *activity itself performed* by the information system. This concerns both decision making activities such as assessing the credit worthiness of a loan applicant, and physical activities, such as assembling two pieces of equipment on a robotic assembly line. In this case, the information system needs to, 1) perform the activity, 2) manage its execution, and 3) report on the result of the activity.

Fig. 11 shows a naïve generic software model for performing activities (abstract model). In this case, we need to specialise the classes **Activity** and **ActivityReport** for each process activity that we wish to support. We illustrated this for the case of credit assessment. A more robust support for

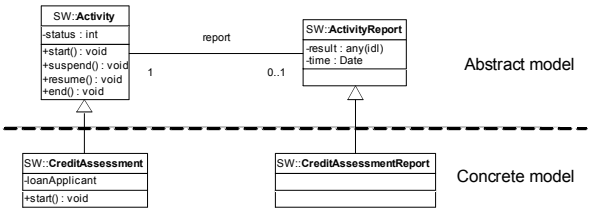

Fig. 11. Model fragment for supporting the software execution of process activities. First-cut.

activity execution could use a variant of the *virtual machine* architectural pattern. Process activities are specified in some *declarative* format and fed into an *activity engine* that *interprets* the activity specification and executes on the input data. This is shown in Fig. 12.

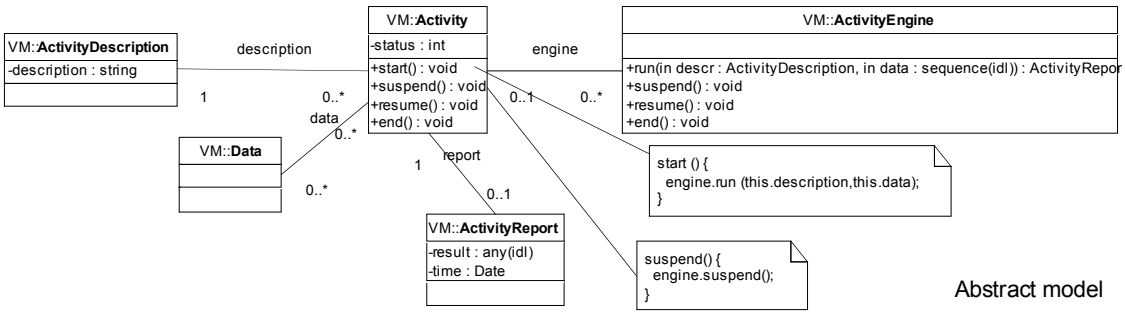

Fig. 12. Supporting the software execution of process activities using the virtual machine style.

In this case, specific activities are represented by *instances* of the model in Fig. 12. An activity is represented by instances of **Activity**. The declarative description of the activity is given in an instance of **ActivityDescription**, and the parameters are given as instances of **Data**. Instances of **Activity** delegate their execution to an **ActivityEngine** which takes the description of the activity and the data, interprets/executes the description, and produces the activity report. The software pattern described in Fig. 12 corresponds to so called business-process management systems (BPM) and business rules management systems (BRMS), and combinations thereof. Process and command control systems also use the same style, as the bulk of the processing is given in the form of command scripts that can be interpreted during run-time.

4.4 Towards a methodology for mapping business process models to software models

A methodology for mapping business process models to software models could proceed as follows:

1. Identify the business processes that we want to support, partially or fully
2. For each such process:
   (a) Determine the scope of automation, in the sense discussed in section 4.3
   (b) For the resulting sub-process, determine the purpose of the information system implementation (simulation vs. automation). This will mostly influence the nature of the mapping to be performed next
   (c) For each activity of the sub-process
      i. Determine the type of automation (recording vs. performing)
      ii. Determine the amount of information to capture (e.g., recording information about actors or not, and how much)
      iii. Perform the mapping
   (d) Compose the model fragments resulting from the individual activities to yield a complete model for the sub-process
3. Compose the process-specific models to yield a complete application model.

For model composition, we can use something along the lines of OORAM's *model synthesis* techniques [11]. Indeed, we can think of activity-specific model fragments as OORAM's *role models* as each model fragment captures a model of a system regarding a particular collaboration. Entities that participate in many activities will end up with the union of the data requirements of the individual activities. The same kind of synthesis applies across processes, where the requirements from different processes are also merged.

5 Discussion

We envision a development process that would enable us to develop information systems by first precisely modeling the business processes that they are meant to support, and then, based on the kind and level of support we wish our information system to provide, generate the corresponding software models. This has been an elusive dream for researchers and practitioners alike, in both software engineering and MIS. There are two major reasons why the mapping from business process to software has resisted formalization. First, we need to have process models with enough precision and detail to reach the level of granularity required for software models. Second, even if we have such models at hand, we need to capture – and formalize – an array of decisions that software planners and analysts make when they develop their systems.

Software development is full of complex transformations that resist formalization The traditional approach to these transformations is to use catalogs of past realizations, i.e., sets of $\langle x, y \rangle$ pairs. To solve a problem P, we look for the $\langle x, y \rangle$ pair where "x" best matches P, and then adapt "y" according to the nature and degree of the match. This approach depends on a good coverage of the problem space. We propose a complementary approach which tries to capture the mapping locally. Our approach should be used when the closest $\langle x, y \rangle$ pair is *beyond* the scope of the adaptation mechanisms.

The work presented in this paper is at embryonic stage. We have started implementing our representation and classification of business processes within the Eclipse environment using the Eclipse Model Framework and extensions thereof to handle the behavioral view. We have barely scratched the surface of the mapping process $\rightarrow$ software. However, we wish to have demonstrated that, with a detailed analysis of the various decision points of this mapping, we are able to capture some of the essence of this mapping.

References

1. W. Carlson. Business information analysis and integration technique (biait) - the new horizon. *Data Base*, 10(4):3–9, 1979.
2. P. Coad, E. Lefebvre, and J. Luca. *Java Modeling In Color With UML: Enterprise Components and Process*. Prentice-Hal, 1999.
3. Workflow Management Coalition. Interface 1: Process definition interchange process model. wfmc-tc-1016-p, version 1.1, October 1999.
4. Workflow Management Coalition. Terminology and glossary. wfmc-tc-1011, February 1999.
5. B. Curtis, M. Kellner, and J. Over. Process modeling. *Com. of the ACM*, 35(9):75– 90, 1992.
6. T. Malone et al. Tools for inventing organizations: Toward a handbook of organizational processes. *Management Science*, 45(3):425–443, March 1999.
7. M. Hammer and J. Champy. *Reengineering the Corporation*. Harper Business, New York, 1993.
8. S. Isoda. Object-oriented real-world modeling revisited. *Journal of Systems and Software*, 59(2):153–162, November 2001.
9. N. Maiden and A. Sutcliffe. Exploiting reusable specifications through analogy. *Com. of the ACM*, 35(4):55–64, April 1992.
10. H. Mili, G. Bou Jaoude, E. Lefebvre, and G. Tremblay. Business process modeling languages: Sorting through the alphabet soup, submitted, 55 p.
11. T. Reenskaug. *Working with Objects: The OORAM Software Engineering Methodology*. Manning, 1996.