Managing business processes as an information resource

by F. Leymann
W. Altenhuber

The relevance of business processes as a major asset of an enterprise is more and more accepted: Business processes prescribe the way in which the resources of an enterprise are used, i.e., they describe how an enterprise will achieve its business goals. Organizations typically prescribe how business processes have to be performed, and they seek information technology that supports these processes. We describe a system that supports the two fundamental aspects of business process management, namely the modeling of processes and their execution. The meta-model of our system deals with models of business processes as weighted, colored, directed graphs of activities; execution is performed by navigation through the graphs according to a well-defined set of rules. The architecture consists of a distributed system with a client/server structure, and stores its data in an object-oriented database system.

Organizations typically prescribe how business processes have to be performed, especially those processes that represent complex routine work, that involve many persons (both concurrently and sequentially), and that are in general frequently performed. Examples of such processes are manifold: program development in the software business, credit allocation in the banking business, customer enrollment in the health insurance business, or expense allowance processing in the administration business.

The relevance of business processes as a major asset of an enterprise is being accepted more and more. Business processes prescribe the way in which the resources (e.g., data, capital, human beings) of an enterprise are used, i.e., they describe how an enterprise will achieve its business goals. The quality of the business processes will influence the quality of the performance of an enterprise. Thus, business processes themselves represent important information resources of an enterprise, and techniques or systems to manage and support business processes are always in demand.

The IBM program product called FlowMark* supports the management of business processes. Both fundamental aspects of process management, namely the modeling of processes (build time) and the execution of processes according to a process model (run time), are facilitated. FlowMark may be perceived especially as a repository for business processes. Within the IBM Information Warehouse* framework (of which some aspects are discussed elsewhere in this issue), FlowMark is positioned as the work flow management component.

Current approaches. Today, there is no generally accepted methodology for modeling business processes. Petri nets are traditionally used to describe and analyze concurrent systems.[1] Nevertheless, it has been recognized that Petri nets are

not currently succinct and manageable enough to become useful in modeling business processes.[2] For this reason high-level Petri nets have been intensively studied during the last couple of years. In particular, predicate/transition nets[3] and colored Petri nets[4] have been applied in various application areas. But other methodologies have also been proposed and applied. It depends on the emphasis one puts on the usage of processes as to whether they are described as Petri nets, trigger systems, forms, case plans, or even collections of formulas of temporal logic, for example.

If one focuses on the pure data manipulation aspects of a process, process models are viewed as vehicles for ensuring database integrity. Guyot[5] shows that Petri nets are allowing database administrators to control and constrain the execution of activities that manipulate a database. Temporal logic has proved to be remarkably successful in describing parallel programs and in studying their properties;[6] the management of parallel components of a program has some similarities to managing transactions concurrently accessing databases. Thus, Lipeck and Saake[7,8] discuss how temporal logic is applied to describe valid sequences of database states and consistency-preserving transactions, which is in certain situations the major intent of a process model. Partial orders on event spaces are also considered to model consistency-preserving sequences of database actions.[9]

Process models may also be perceived as a means to extend and complement facilities known from conventional transaction processing monitors. A process model is viewed as the specification of the flow of control and the flow of data separate from the collection of routines performing the proper computations of an individual application. Applications are represented in Garcia-Molina and Salem[10] simply as sequences of related transactions ensuring either the successful execution of all transactions in the chain or its compensation. Predicate/transition nets are pursued in Wächter and Reuter[11] to model networks of "steps" (and "compensation steps") that represent the "scripts" of an application. A network of "activities," which are triggered by "events," which are sent along "arcs" connecting activities, is exploited in Hsu et al.[12] for that purpose. A very generic and abstract approach ("spheres of control") especially for managing flows, including their recovery, is presented in Davis.[13] Targeting in the modeling of long transactions, Kotz[9] proposes the use of event/trigger systems.

With the support of office work in mind the following has been suggested: To model office procedures, Behrmann-Poitiers and Edelmann[14] are proposing "case plans." In cases where the office system reveals specific object-oriented characteristics, life-cycle diagrams and composed activities are pursued.[15] If a process can be described as the processing of a form, a corresponding proposal is given by Tsichritzis.[16]

The commonality among processes in the areas of software development,[17–19] office work, and administration has been worked out by Chroust and Leymann.[20] A methodology applicable in these areas is described by Leymann.[21] It strives especially for a formalization of processes and their models in these problem areas that is even more succinct and "user-friendly" than in Genrich[3] and Jensen.[4] It encompasses, for example, the case plans of Behrmann-Poitiers and Edelmann[14] and in addition allows the definition of parameter-controlled work flows in their problem domain. In this paper the methodology of Leymann[21] is enhanced by introducing *PM-graphs* (Process Model graphs) in order to fulfill additional requirements posed by FlowMark.

Our presentation. First, we show that today process models are treated as information resources in a rudimentary manner; also, we sketch the potential embedding of process models into an IRDS (information resource dictionary system[22]). Next, we discuss the meta-model of FlowMark for processes. We provide and motivate a collection of constructs that have to be used in order to define the model of a process to FlowMark. Then, the architecture of FlowMark is sketched. It is a distributed system with a client/server system structure. All relevant data are stored in an object-oriented database system. The graphical end-user interfaces for defining and executing processes are sketched. Activities represented by executables complying to a certain invocation paradigm are invoked by FlowMark. In the final section, we give a mathematical formulation of the syntax and the semantics of the meta-model. Mathematically, a process model is represented as a special weighted, colored, directed graph of activities (called a PM-graph); the semantics of the meta-model are defined operationally, exe-

cuting a process as an instance of a process model by navigating through the PM-graph according to a well-defined set of rules.

Process models as information resources

Along with the classical production factors of land, labor, and capital, enterprises are considering information more and more as an important resource, i.e., as one of their assets. This information includes data about all resources needed to reach the goal of the enterprise. It is generally accepted that this information should be represented in a formal manner as much as possible. The collection of actions needed to achieve this goal is referred to as "enterprise modeling."

Enterprise models. The conceptual base for enterprise modeling is sometimes called a *hypersemantic data model*.[23] Producing the model of a concrete enterprise by using a hypersemantic data model results in an *enterprise model*. Such an enterprise model consists of two components: the *data model* and the *knowledge model*.[23]

The data model describes the structure of all resources of the enterprise and is thus somewhat like the syntactical component of the enterprise model; in this sense, the data model describes *what* can be used by the enterprise to reach its goal. Today, enterprises are building data models based on *semantic data models* (for example, the entity/relationship model; for an overview of different semantic data models see Peckham and Maryanski[24]).

The knowledge model describes the use of resources and their connections; it is the semantical component of the enterprise model. In this sense, the knowledge model describes *how* the enterprise uses its resources in order to reach its goal. The knowledge model encompasses constraints, heuristics, and procedures. Constraints define the local and global consistency of the resources; if the resources are stored in a database, constraints fix the valid database states. Heuristics describe how to derive data. Procedures define events and correlated actions, set sequences of actions, and describe business processes. The model of a business process can be used to define, for example, valid sequences of state transitions in a database (intertransaction integrity[7,8,11]) as well as sequences of work steps, whereas executing a business process could be described as context-dependent.[17,20]

Documents as process models. Process models represent knowledge about an enterprise. Usually, this knowledge is found today in the format of textual processing instructions. The handling of each single business process according to these processing instructions then corresponds to a process, i.e., an instance of the process model as defined via the instructions.

The knowledge model describes the use of resources and their connections.

Process models in the format of textual processing instructions are very inflexible. Enterprises cannot react with sufficient speed to changes in their environment. Changes in the processing instructions are communicated by distributing documents, thus, the time to activate these changes is dependent on when the corresponding documents arrive; in distributed organizations these documents will arrive at different points in time, resulting, in turn, in consistency problems. Support for handling individual processes is only enabled in a limited way via the textual processing instructions. Precise compliance with the instructions cannot be enforced directly.

Control programs as process models. Today, computer assistance for handling processes is achieved, for example, by programming the corresponding procedural instructions. Each single work step can be supported via "generic programs" (i.e., via tools or service routines), via special applications (i.e., via programs considering the individual needs of an enterprise), or simply via help texts (i.e., via electronically documented work instructions). A special control program determines the individual sequence of work steps dependent on the concrete context of the individual business process. In this sense, the control program represents the formal-

ized process model, and an instance of the control program corresponds to a concrete, individual process. But such control programs also do not allow for highly dynamic reactions. Changing the knowledge embedded in the control program involves a great deal of effort (redesign, coding, compiling, etc.).

A process model may be seen as a template for a class of similar business processes performed within an enterprise.

Separate representations for process models. For this reason, extracting the knowledge representing the process from the control program and forming a separate representation of this knowledge as its own syntactical unit is extremely desirable. As a consequence, these syntactical units—which now represent the process model—have the flexibility and comprehensibility to institute the required dynamics. A process interpreter receiving such a process model as input (along with other information) can instantiate the process model and determine the individual sequence of work steps, depending on the context of the instantiation of the process model.[20]

IRDS and process models. Process *models* describe (apparently) different functions such as the production of a part in a production line, the settlement of a damage case within an insurance company, the treatment of a form for making allowances for expenses, the procedure in developing a program, or valid sequences of transactions. In these situations, each individually executing process can be perceived as a separate *instance* of the process model.[20] The process model is the processing instruction for a concrete process to be executed (and is thus a processing model). Computer assistance thus means both the support for defining the process model (*modeling*) and the support for performing each individual process (*execution*).

Computer assistance for defining process models should be enabled via a language that provides constructs that can be embedded canonically in a dictionary. A dictionary concept that strives for integrating data models and process models is pursued and thus contains *all* information resources. It follows the International Organization for Standardization (ISO) conceptualization principle,[25] according to which as much knowledge about an application area as possible is moved from the programs to the dictionary of an enterprise. Such a language then encompasses a model for process models, i.e., a *meta-model* for processes; instances of the meta model are process models. In turn, the instances of the process models are the proper representations of process executions. Meta-model and process model, and process model and process, respectively, are building intension-, extension-pairs[26] that thus comply conceptually to the ISO Dictionary Architecture IRDS.[22] The integration of process models into the IRDS together with the already available data modeling capabilities then allows very flexible enterprise modeling based on a dictionary. For that purpose one has to describe our meta-model in terms of the fundamental modeling language of the IRDS.

FlowMark allows process models to be defined according to our meta-model described below. Each process model is an instance of the meta model. Process models are instantiated and executed by interpreting the instances of the types of the meta-model. The interpretation is performed by navigating through each individual instance (process) in accordance with the underlying process model.

The meta-model

A process model may be seen as a template for a class of similar business processes performed within an enterprise. It is a schema describing all possible variants of the (dynamic!) execution of a particular kind of business process. Each individual process is an instance of a process model, and it represents a concrete, specific execution of a variant prescribed by the process model.

The fundamental building block of the meta-model is the *activity*. An activity represents a business action that is a semantical unit at a certain phase of modeling effort. It might have a fine-structure, which is then represented in turn via a

process model, or the details of it might not be of interest at all from a modeling perspective. It is important to note that these fine-structures allow both a bottom-up and a top-down approach to process modeling.

As an example, suppose a process model for credit allocation contains an activity called Solvency. For the modeler of credit allocation it is not of interest *how* Solvency is checked, but rather to make sure that this check *will* take place. The refinement of the activity Solvency as a process model again (if required) can be done (or may have already been done) by a different modeler.

In general, the work represented by an activity produces results. Within the meta-model the types of results of this work are associated with the activity as parameter types. Now, activities generally access types of results of other activities, or require information about the context of the current activity; such parameter types can also be associated with an activity. In general, an activity is associated with both *input parameter types* and *output parameter types* (in cases in which no misunderstanding will occur, the suffix "type" is omitted).

The collection of all input parameters of an activity is referred to as the *input container* of that activity, and the collection of all of its output parameters is referred to as the *output container*. Since process models may serve as fine-structures of activities, each process model itself is associated with both an input container and an output container; note that the input or output container of a process provides some sort of "global context" for all activities contained within this process. A concrete execution of an activity (also called an *activity instance*) is thus accessing the instances of the input parameter types from its input container and will produce instances of the output parameter types from its output container. Because of this, activities are considered to be mathematical maps.

In practice, only the "process-relevant" parameters of an activity are explicitly defined (i.e., externalized) rather than all parameters affected by an execution of the activity. For example, an activity generally modifies data that are not defined in its associated containers because these data are not of interest to other activities within the process; or an activity might obtain (additional) input from sources different from its input container (e.g., database reads).

As a result, it is pragmatic to recommend capturing an activity as a *relation* between its input container and its output container (for example, because additional input as mentioned before might result in nondeterministic behavior of the activity with respect to its input container). In fact, choosing whether activities are "maps" instead of "relations" is not crucial to our meta-model, and the model could be easily made to accommodate a choice. Nevertheless, for simplicity we treat activities as maps because we consider this treatment to be more suited to the perception of process modelers.

In general, the activities of a process may not be executed in an arbitrary manner. Some activities are necessary for a process to start, some activities might only be run when others are finished, and so on. In other words, the activities of a process form a network with arcs that point from a given activity to its successor activities. Since a process model has to reflect all possible valid executions of a specific business procedure, each activity within a process model must be connected to all of its potential follow-on activities. A process model may be perceived as a directed graph having nodes that are the activities of the process and having edges that connect an activity with its potential successors. Since an edge represents the potential control flow from one activity to another, it is also referred to as a *control connector*.

As an example, suppose the credit allocation process model contains the activities Solvency, Reject, Accept, BranchManagerApproval, and Notify. The potential follow-on activities of Solvency are Reject, Accept, and BranchManagerApproval. BranchManagerApproval has the potential successors Reject and Accept. The activity Notify is the successor of both Reject and Accept.

When a process model is executed (or *instantiated*) it depends on the concrete situation in which the process is run as to what subset of the set of all potential follow-on activities of a particular activity is really executed once this particular activity is terminated successfully. A "concrete situation" is captured by the collection

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.